Dataset import

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.metrics import mean_absolute_error, mean_squared_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from sklearn.preprocessing import MinMaxScaler
import warnings
warnings.filterwarnings("ignore")

# Load dataset
df = pd.read_csv("C:\\Users\\marat\\OneDrive\\Desktop\\project\\global
temperature.csv")

df['Date'] = pd.to_datetime(df[['Year', 'Month', 'Day']])
df.set_index('Date', inplace=True)
df = df[['Temperature', 'Anomaly']]

 # Resample to monthly data
df.tail()
```

Date	Temperature	Anomaly
2022-07-27	10.229	1.639
2022-07-28	10.221	1.631
2022-07-29	10.164	1.574
2022-07-30	10.167	1.577
2022-07-31	10.219	1.629

Descriptive Statistics

```
import pandas as pd

df = pd.read_csv(r"C:\Users\marat\OneDrive\Desktop\project\global
temperature.csv")
# Keep only relevant columns
df1 = df[['Year', 'Month', 'Temperature']]

# Descriptive statistics for each year
yearly_stats = df1.groupby('Year')['Temperature'].describe()
```

```
# Descriptive statistics for each month across all years
monthly_stats = df1.groupby('Month')['Temperature'].describe()

# Grouping into specific 20-year periods
bins = list(range(df1['Year'].min(), df1['Year'].max() + 1, 20))
bin_labels = [f"{bins[i]}-{bins[i+1]-1}" for i in range(len(bins)-1)]
df1['Year Group'] = pd.cut(df1['Year'], bins=bins, labels=None) #
Changed to None to avoid label issues
grouped_20yr_stats = df1.groupby('Year Group')
['Temperature'].describe()

# Splitting the dataset into two equal time periods
mid_year = df1['Year'].median()
df1['Half'] = df1['Year'].apply(lambda x: 'First Half' if x <=
mid_year else 'Second Half')
half_stats = df1.groupby('Half')['Temperature'].describe()

# Overall statistics
overall_stats = df1['Temperature'].describe()

# Compute average temperature for each month of each year
monthly_avg_per_year = df1.groupby(['Year', 'Month'])
['Temperature'].mean().unstack()

# Compute average temperature for each year
yearly_avg_temp = df1.groupby('Year')['Temperature'].mean()

# Calculate 10-year averages
bins_10yr = list(range(df1['Year'].min(), df1['Year'].max() + 1, 10))
bin_labels_10yr = [f"{bins_10yr[i]}-{bins_10yr[i+1]-1}" for i in
range(len(bins_10yr)-1)]
df1['Year Group 10'] = pd.cut(df1['Year'], bins=bins_10yr,
labels=bin_labels_10yr, right=False)
grouped_10yr_avg = df1.groupby('Year Group 10')['Temperature'].mean()

# ---- Descriptive Statistics for 10-Year Groups ----
bins_10yr = list(range(df1['Year'].min(), df1['Year'].max() + 10, 10))
labels_10yr = [f"{bins_10yr[i]}-{bins_10yr[i+1]-1}" for i in
range(len(bins_10yr)-1)]
df1['Year Group 10'] = pd.cut(df1['Year'], bins=bins_10yr,
labels=labels_10yr, right=False)
grouped_10yr_stats = df1.groupby('Year Group 10')
['Temperature'].describe()

# ---- Descriptive Statistics for 20-Year Groups ----
bins_20yr = list(range(df1['Year'].min(), df1['Year'].max() + 20, 20))
labels_20yr = [f"{bins_20yr[i]}-{bins_20yr[i+1]-1}" for i in
range(len(bins_20yr)-1)]
df1['Year Group 20'] = pd.cut(df1['Year'], bins=bins_20yr,
labels=labels_20yr, right=False)
```

```
grouped_20yr_stats = df1.groupby('Year Group 20')
['Temperature'].describe()

# ---- Descriptive Statistics for 35-Year Groups ----
bins_35yr = list(range(df1['Year'].min(), df1['Year'].max() + 35, 35))
labels_35yr = [f"{bins_35yr[i]}-{bins_35yr[i+1]-1}" for i in
range(len(bins_35yr)-1)]
df1['Year Group 35'] = pd.cut(df1['Year'], bins=bins_35yr,
labels=labels_35yr, right=False)
grouped_35yr_stats = df1.groupby('Year Group 35')
['Temperature'].describe()

# Save results to an Excel workbook
with pd.ExcelWriter("temperature_analysis_new1.xlsx") as writer:
    yearly_stats.to_excel(writer, sheet_name="Yearly Statistics")
    monthly_stats.to_excel(writer, sheet_name="Monthly Statistics")
    grouped_20yr_stats.to_excel(writer, sheet_name="20-Year Groups")
    half_stats.to_excel(writer, sheet_name="Two Halves")
    overall_stats.to_excel(writer, sheet_name="Overall Statistics")
    monthly_avg_per_year.to_excel(writer, sheet_name="Monthly Avg Per
Year")
    yearly_avg_temp.to_excel(writer, sheet_name="Yearly Avg Temp")
    grouped_10yr_avg.to_excel(writer, sheet_name="10-Year Averages")
    grouped_10yr_stats.to_excel(writer, sheet_name="10-Year Stats")
    grouped_20yr_stats.to_excel(writer, sheet_name="20-Year Stats")
    grouped_35yr_stats.to_excel(writer, sheet_name="35-Year Stats")
# Save 10-year averages

# Display results
selected_years = list(range(1880, 2021, 10))
print("Descriptive Statistics by Selected Years (Every 10 Years):\n",
yearly_stats.loc[selected_years])
print("\nDescriptive Statistics by Month:\n", monthly_stats)
print("\nDescriptive Statistics by 20-Year Groups:\n",
grouped_20yr_stats)
print("\nDescriptive Statistics for Two Equal Halves:\n", half_stats)
print("\nOverall Statistics:\n", overall_stats)
print("\nMonthly Average Temperature per Year (First 5 Years):\n",
monthly_avg_per_year.head())
print("\nYearly Average Temperature (First 5 Years):\n",
yearly_avg_temp.head())
print("\n10-Year Averages:\n", grouped_10yr_avg) # Display 10-year
averages
print("\nDescriptive Statistics by 10-Year Groups:\n",
grouped_10yr_stats)
print("\nDescriptive Statistics by 20-Year Groups:\n",
grouped_20yr_stats)
print("\nDescriptive Statistics by 35-Year Groups:\n",
grouped_35yr_stats)
```

```
Descriptive Statistics by Selected Years (Every 10 Years):
       count       mean       std    min      25%     50%       75%
max
Year

1880  366.0   7.995557  0.370984  6.945  7.76175   8.063   8.29675
8.666
1890  365.0   7.989942  0.300809  6.933  7.77400   8.003   8.20300
8.770
1900  365.0   8.528134  0.426486  6.916  8.34500   8.572   8.82600
9.367
1910  365.0   8.211362  0.395085  6.751  8.03100   8.280   8.47400
8.974
1920  366.0   8.305962  0.436845  6.956  8.14125   8.366   8.55550
9.524
1930  365.0   8.508871  0.430060  6.978  8.26500   8.479   8.74800
9.909
1940  366.0   8.680478  0.345768  7.759  8.48550   8.681   8.92300
9.566
1950  365.0   8.292293  0.390372  7.121  8.06000   8.355   8.58000
9.142
1960  366.0   8.484902  0.500478  7.228  8.14100   8.556   8.79875
9.672
1970  365.0   8.636427  0.397900  7.476  8.38200   8.619   8.88500
9.854
1980  366.0   8.936541  0.410590  7.656  8.68350   8.962   9.17225
10.136
1990  365.0   9.236841  0.530523  8.025  8.87500   9.169   9.56300
10.847
2000  366.0   9.169997  0.490043  8.017  8.83100   9.166   9.44050
10.752
2010  365.0   9.713625  0.471451  8.292  9.40800   9.693  10.00400
10.958
2020  366.0  10.058536  0.431781  9.002  9.78600  10.032  10.29725
11.327

Descriptive Statistics by Month:
         count      mean       std    min      25%    50%      75%
max
Month

1      4433.0  8.606167  0.836344  5.862  8.00400  8.578  9.17500
11.055
2      4039.0  8.622804  0.842865  6.117  8.00800  8.539  9.18250
11.545
3      4433.0  8.738859  0.826366  6.821  8.13500  8.625  9.26800
11.491
4      4290.0  8.902750  0.690573  7.156  8.40825  8.817  9.33800
11.352
5      4433.0  8.764846  0.591479  7.246  8.35600  8.668  9.10100
```

```
10.924
6      4290.0  8.720883  0.542304  7.567  8.33700  8.615  9.00475
10.518
7      4433.0  8.742596  0.476396  7.612  8.41300  8.633  8.95600
10.499
8      4402.0  8.675235  0.506688  7.594  8.32125  8.554  8.93800
10.566
9      4260.0  8.628015  0.538040  7.448  8.25075  8.509  8.92000
10.351
10     4402.0  8.754561  0.586063  7.054  8.34200  8.695  9.09600
10.623
11     4260.0  8.526160  0.674768  6.751  8.05900  8.454  8.94225
10.945
12     4402.0  8.572129  0.740657  6.412  8.07400  8.489  9.06275
11.020

Descriptive Statistics by 20-Year Groups:
                 count      mean       std    min    25%    50%
75%  \
Year Group 20

1880-1899      7305.0  8.112854  0.433809  5.862  7.881  8.163
8.4010
1900-1919      7304.0  8.258179  0.431921  6.412  8.015  8.286
8.5390
1920-1939      7305.0  8.478008  0.434693  6.597  8.245  8.482
8.7220
1940-1959      7305.0  8.594258  0.433365  6.803  8.341  8.622
8.8690
1960-1979      7305.0  8.582274  0.458677  6.596  8.290  8.581
8.8860
1980-1999      7305.0  9.025537  0.502520  7.257  8.694  9.036
9.3560
2000-2019      7305.0  9.606187  0.503268  7.906  9.283  9.602
9.9210
2020-2039       943.0  9.944686  0.410681  8.162  9.674  9.916
10.1725

                   max
Year Group 20
1880-1899        9.583
1900-1919        9.704
1920-1939       10.209
1940-1959       10.152
1960-1979       10.282
1980-1999       10.847
2000-2019       11.545
2020-2039       11.352

Descriptive Statistics for Two Equal Halves:
```

```
                count      mean       std    min    25%    50%    75%
max
Half

First Half   26297.0  8.337875  0.469145  5.862  8.070  8.363  8.633
10.209
Second Half  25780.0  9.046162  0.661319  6.596  8.567  9.011  9.503
11.545

Overall Statistics:
 count    52077.000000
mean         8.688503
std          0.673085
min          5.862000
25%          8.250000
50%          8.603000
75%          9.088000
max         11.545000
Name: Temperature, dtype: float64

Monthly Average Temperature per Year (First 5 Years):
 Month         1         2         3         4         5         6
7   \
Year

1880   7.690903  7.337310  7.564581  7.773167  8.208258  8.272533
8.381677
1881   8.209839  8.072500  8.336581  8.603800  8.487452  8.023933
8.467903
1882   8.879806  8.641679  8.333516  8.068467  8.179645  8.012533
8.102613
1883   7.413419  7.409107  8.157935  8.131200  8.213290  8.495900
8.212645
1884   8.143226  7.980586  7.422839  7.490367  7.503839  7.783433
7.944032

Month         8         9         10        11        12
Year
1880   8.314419  7.988467  8.050000  8.178300  8.152032
1881   8.531065  8.223300  8.074387  7.776267  8.227097
1882   8.354968  8.234467  7.841419  7.505133  7.274097
1883   7.983774  7.902900  8.142387  7.647100  8.181710
1884   8.070097  7.807233  7.857484  7.721533  7.896355

Yearly Average Temperature (First 5 Years):
 Year
1880    7.995557
1881    8.255378
1882    8.116529
1883    7.995145
```

```
1884    7.801880
Name: Temperature, dtype: float64

10-Year Averages:
 Year Group 10
1880-1889    8.051050
1890-1899    8.174676
1900-1909    8.260697
1910-1919    8.255661
1920-1929    8.416542
1930-1939    8.539491
1940-1949    8.647585
1950-1959    8.540916
1960-1969    8.553706
1970-1979    8.610850
1980-1989    8.896096
1990-1999    9.155014
2000-2009    9.485445
2010-2019    9.726963
Name: Temperature, dtype: float64

Descriptive Statistics by 10-Year Groups:
                 count      mean       std    min      25%     50%
75%  \
Year Group 10

1880-1889      3653.0  8.051050  0.423024  6.529  7.81400  8.0750
8.33200
1890-1899      3652.0  8.174676  0.435706  5.862  7.96700  8.2420
8.45125
1900-1909      3652.0  8.260697  0.427165  6.662  8.01400  8.2800
8.53200
1910-1919      3652.0  8.255661  0.436670  6.412  8.01500  8.2905
8.54700
1920-1929      3653.0  8.416542  0.420882  6.597  8.21600  8.4300
8.63900
1930-1939      3652.0  8.539491  0.439623  6.793  8.28775  8.5540
8.79300
1940-1949      3653.0  8.647585  0.400426  6.803  8.42800  8.6640
8.89700
1950-1959      3652.0  8.540916  0.457861  7.024  8.25400  8.5675
8.83400
1960-1969      3653.0  8.553706  0.455693  6.596  8.25200  8.5620
8.85100
1970-1979      3652.0  8.610850  0.459933  6.610  8.32675  8.5980
8.91000
1980-1989      3653.0  8.896096  0.470858  7.352  8.57400  8.9000
9.23200
1990-1999      3652.0  9.155014  0.499890  7.257  8.83775  9.1560
9.45225
```

```
2000-2009      3653.0  9.485445  0.483732  7.906  9.17200  9.4830
9.79800
2010-2019      3652.0  9.726963  0.493415  8.156  9.41400  9.7050
10.02700
2020-2029       943.0  9.944686  0.410681  8.162  9.67400  9.9160
10.17250

                   max
Year Group 10
1880-1889        9.550
1890-1899        9.583
1900-1909        9.704
1910-1919        9.671
1920-1929       10.209
1930-1939       10.166
1940-1949       10.015
1950-1959       10.152
1960-1969       10.282
1970-1979       10.007
1980-1989       10.506
1990-1999       10.847
2000-2009       11.142
2010-2019       11.545
2020-2029       11.352

Descriptive Statistics by 20-Year Groups:
                 count      mean       std    min    25%    50%
75%  \
Year Group 20

1880-1899      7305.0  8.112854  0.433809  5.862  7.881  8.163
8.4010
1900-1919      7304.0  8.258179  0.431921  6.412  8.015  8.286
8.5390
1920-1939      7305.0  8.478008  0.434693  6.597  8.245  8.482
8.7220
1940-1959      7305.0  8.594258  0.433365  6.803  8.341  8.622
8.8690
1960-1979      7305.0  8.582274  0.458677  6.596  8.290  8.581
8.8860
1980-1999      7305.0  9.025537  0.502520  7.257  8.694  9.036
9.3560
2000-2019      7305.0  9.606187  0.503268  7.906  9.283  9.602
9.9210
2020-2039       943.0  9.944686  0.410681  8.162  9.674  9.916
10.1725

                   max
Year Group 20
1880-1899        9.583
```

```
1900-1919       9.704
1920-1939      10.209
1940-1959      10.152
1960-1979      10.282
1980-1999      10.847
2000-2019      11.545
2020-2039      11.352

Descriptive Statistics by 35-Year Groups:
                   count      mean       std    min    25%    50%
75%  \
Year Group 35

1880-1914      12783.0  8.177261  0.436764  5.862  7.935  8.215
8.4610
1915-1949      12784.0  8.492936  0.445525  6.412  8.251  8.511
8.7600
1950-1984      12784.0  8.612535  0.474145  6.596  8.308  8.613
8.9140
1985-2019      12783.0  9.378631  0.566654  7.257  9.012  9.378
9.7480
2020-2054        943.0  9.944686  0.410681  8.162  9.674  9.916
10.1725

                   max
Year Group 35
1880-1914       9.704
1915-1949      10.209
1950-1984      10.282
1985-2019      11.545
2020-2054      11.352
```

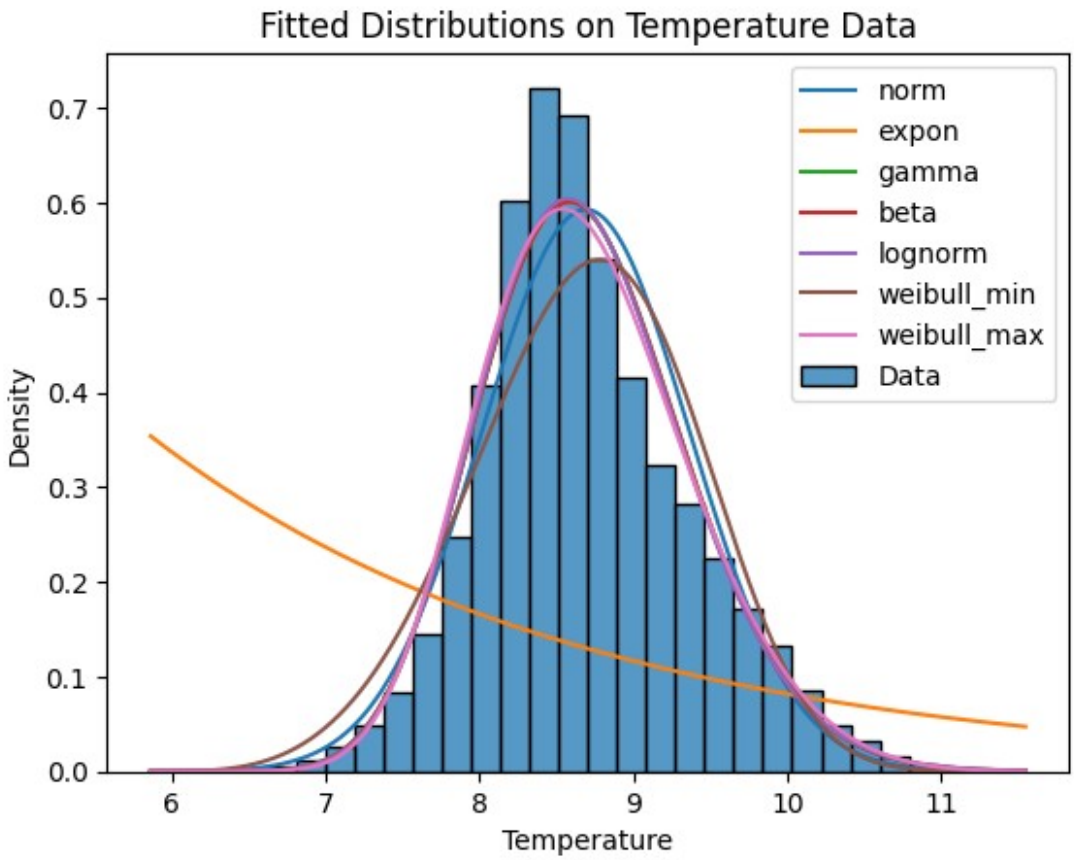
```python
# ---- Descriptive Statistics for 10-Year Groups ----
bins_10yr = list(range(df1['Year'].min(), df1['Year'].max() + 10, 10))
labels_10yr = [f"{bins_10yr[i]}-{bins_10yr[i+1]-1}" for i in
range(len(bins_10yr)-1)]
df1['Year Group 10'] = pd.cut(df1['Year'], bins=bins_10yr,
labels=labels_10yr, right=False)
grouped_10yr_stats = df1.groupby('Year Group 10')
['Temperature'].describe()

# ---- Descriptive Statistics for 20-Year Groups ----
bins_20yr = list(range(df1['Year'].min(), df1['Year'].max() + 20, 20))
labels_20yr = [f"{bins_20yr[i]}-{bins_20yr[i+1]-1}" for i in
range(len(bins_20yr)-1)]
df1['Year Group 20'] = pd.cut(df1['Year'], bins=bins_20yr,
labels=labels_20yr, right=False)
grouped_20yr_stats = df1.groupby('Year Group 20')
['Temperature'].describe()
```

```
# ---- Descriptive Statistics for 35-Year Groups ----
bins_35yr = list(range(df1['Year'].min(), df1['Year'].max() + 35, 35))
labels_35yr = [f"{bins_35yr[i]}-{bins_35yr[i+1]-1}" for i in
range(len(bins_35yr)-1)]
df1['Year Group 35'] = pd.cut(df1['Year'], bins=bins_35yr,
labels=labels_35yr, right=False)
grouped_35yr_stats = df1.groupby('Year Group 35')
['Temperature'].describe()

with pd.ExcelWriter("temperature_analysis_new.xlsx") as writer:
    yearly_stats.to_excel(writer, sheet_name="Yearly Statistics")
    monthly_stats.to_excel(writer, sheet_name="Monthly Statistics")
    grouped_20yr_stats.to_excel(writer, sheet_name="20-Year Groups")
    half_stats.to_excel(writer, sheet_name="Two Halves")
    overall_stats.to_excel(writer, sheet_name="Overall Statistics")
    monthly_avg_per_year.to_excel(writer, sheet_name="Monthly Avg Per
Year")
    yearly_avg_temp.to_excel(writer, sheet_name="Yearly Avg Temp")
    grouped_10yr_avg.to_excel(writer, sheet_name="10-Year Averages") #
Save 10-year averages
```

Distributional Study

```
from scipy.stats import kstest, probplot
from scipy import stats

dist_names = ['norm', 'expon', 'gamma', 'beta', 'lognorm',
'weibull_min', 'weibull_max']
temperature_data = df['Temperature'].dropna()

best_fit = {}
kstest_results = {}

def evaluate_fit(data, dist_name, params):
    dist = getattr(stats, dist_name)
    ks_stat, ks_pval = kstest(data, dist_name, args=params)
    return ks_stat, ks_pval

for dist_name in dist_names:
    dist = getattr(stats, dist_name)
    params = dist.fit(temperature_data)
    best_fit[dist_name] = params
    ks_stat, ks_pval = evaluate_fit(temperature_data, dist_name,
params)
    kstest_results[dist_name] = {'KS Statistic': ks_stat, 'P-Value':
ks_pval}

# Convert results to a DataFrame
```

```python
kstest_df = pd.DataFrame.from_dict(kstest_results, orient='index')

# Plot histogram with fitted distributions
sns.histplot(temperature_data, kde=False, bins=30, stat="density",
label="Data")

x = np.linspace(temperature_data.min(), temperature_data.max(), 1000)

for dist_name, params in best_fit.items():
    dist = getattr(stats, dist_name)
    pdf_fitted = dist.pdf(x, *params)
    plt.plot(x, pdf_fitted, label=dist_name)

plt.legend()
plt.xlabel("Temperature")
plt.ylabel("Density")
plt.title("Fitted Distributions on Temperature Data")
plt.show()



# Print KS Test results
print("\nKolmogorov-Smirnov Test Results:")
print(kstest_df)

# Print best-fit parameters for each distribution
print("\nBest Fit Distribution Parameters:")
for dist_name, params in best_fit.items():
    print(f"{dist_name}: {params}")
```

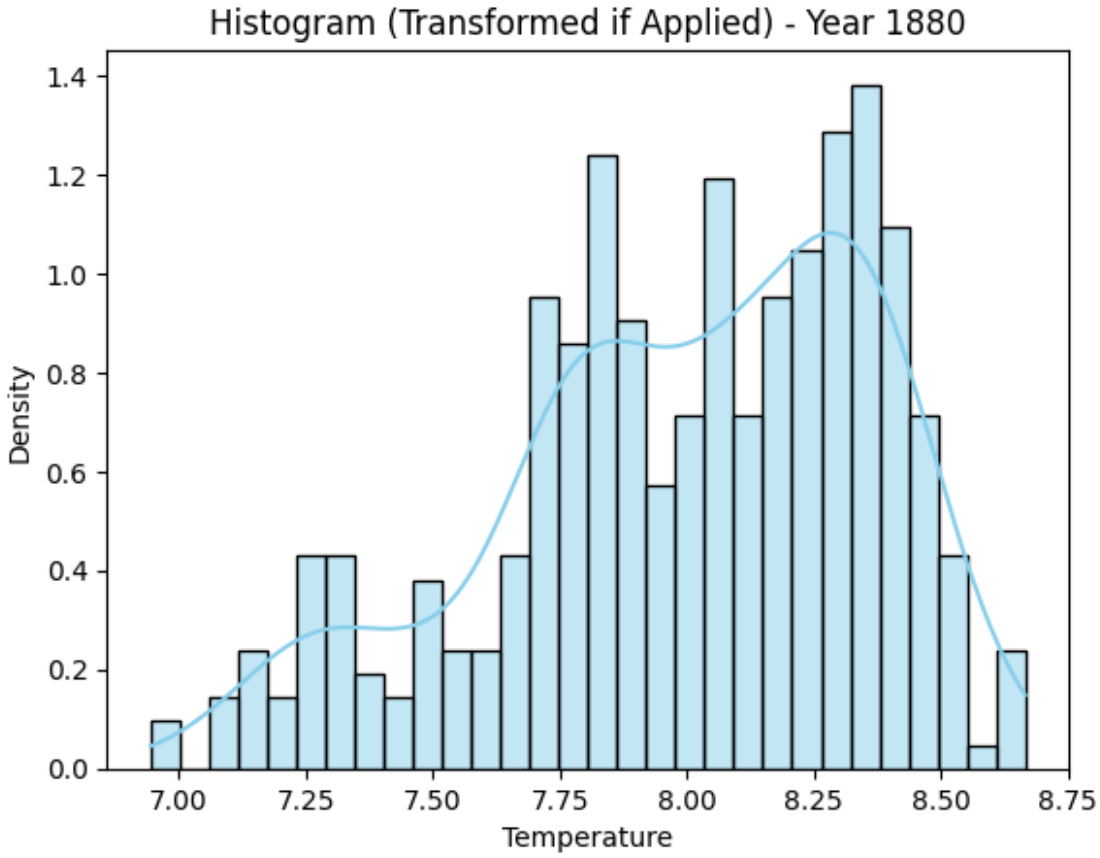


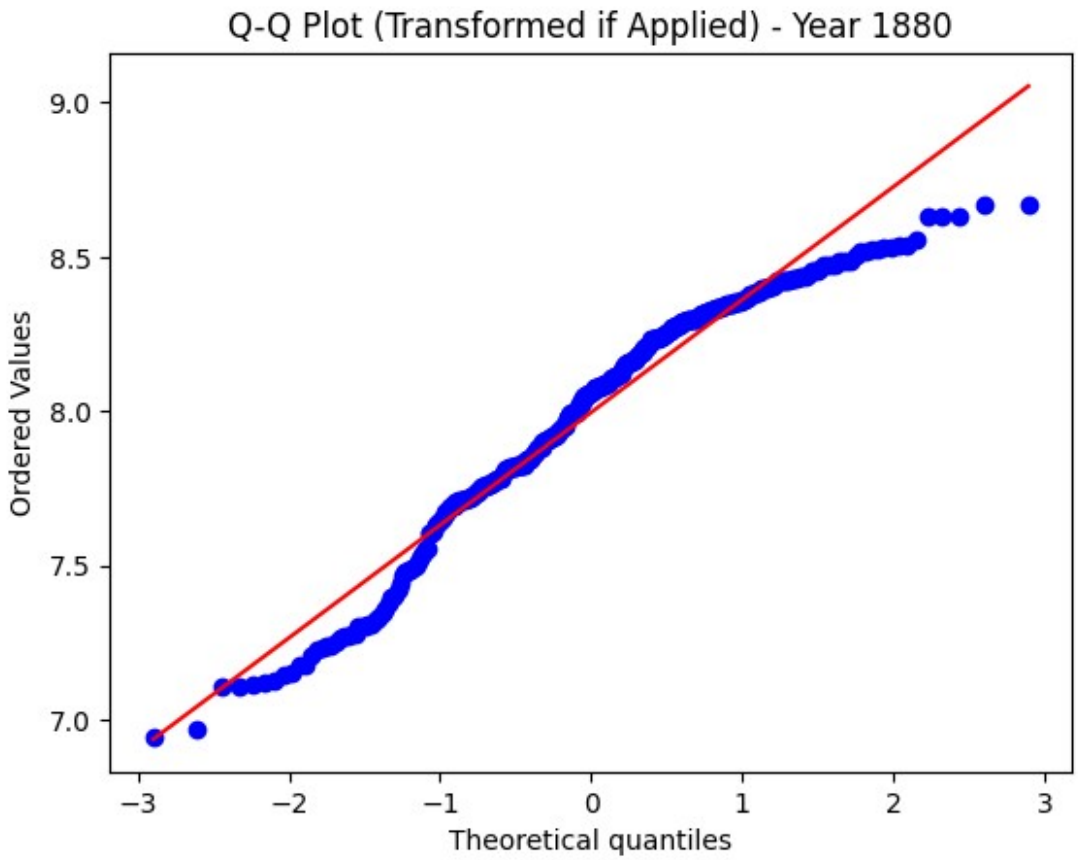

```
Kolmogorov-Smirnov Test Results:
             KS Statistic        P-Value
norm             0.059938  4.460529e-163
expon            0.427401   0.000000e+00
gamma            0.039049   1.968802e-69
beta             0.039049   1.967192e-69
lognorm          0.037038   1.654976e-62
weibull_min      0.074443  1.982387e-251
weibull_max      0.036362   2.928558e-60

Best Fit Distribution Parameters:
norm: (8.688502563511724, 0.6730782024854579)
expon: (5.862, 2.826502563511724)
gamma: (39.011663603959136, 4.497700761300052, 0.10742432871731672)
beta: (39.011788306791566, 47129810.30498446, 4.497692042424953,
5062886.853759905)
lognorm: (0.11628643083088369, 2.9328391979333386, 5.716837978733398)
weibull_min: (4.415801985131122, 5.859482916670485, 3.093586658959958)
weibull_max: (5.760986512463964, 12.050427154466213,
3.6289129305249883)
```

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
from scipy.stats import shapiro

# List of years to test
target_years = [1880, 1900, 1925, 1950, 1975, 2000, 2020]

# Iterate through each year
for year in target_years:
    year_data = df[df['Year'] == year]['Temperature'].dropna()

    if len(year_data) < 3:
        print(f"Year {year}: Not enough data for testing.")
        continue

    print(f"\n===== Year {year} =====")

    # Normality test (Shapiro-Wilk is good for small to medium sample
sizes)
    stat, p_value = shapiro(year_data)
    print(f"Shapiro-Wilk test statistic: {stat:.4f}, p-value:
{p_value:.4f}")

    if p_value > 0.05:
        print("Data appears to be normally distributed.\n")
        transformed_data = year_data  # No need to transform
    else:
        print("Data is NOT normally distributed. Applying Johnson
transformation...\n")

        # Fit Johnson SU distribution
        fitted_params = stats.johnsonsu.fit(year_data)
        transformed_data =
stats.johnsonsu(*fitted_params).ppf(stats.johnsonsu(*fitted_params).cd
f(year_data))

        # Re-test for normality after transformation
        stat_t, p_t = shapiro(transformed_data)
        print(f"Post-transformation Shapiro-Wilk: stat = {stat_t:.4f},
p = {p_t:.4f}")

    # Plot the final data (original or transformed)
    sns.histplot(transformed_data, bins=30, kde=True, stat="density",
color='skyblue')
    plt.title(f"Histogram (Transformed if Applied) - Year {year}")
    plt.xlabel("Temperature")
    plt.ylabel("Density")
```

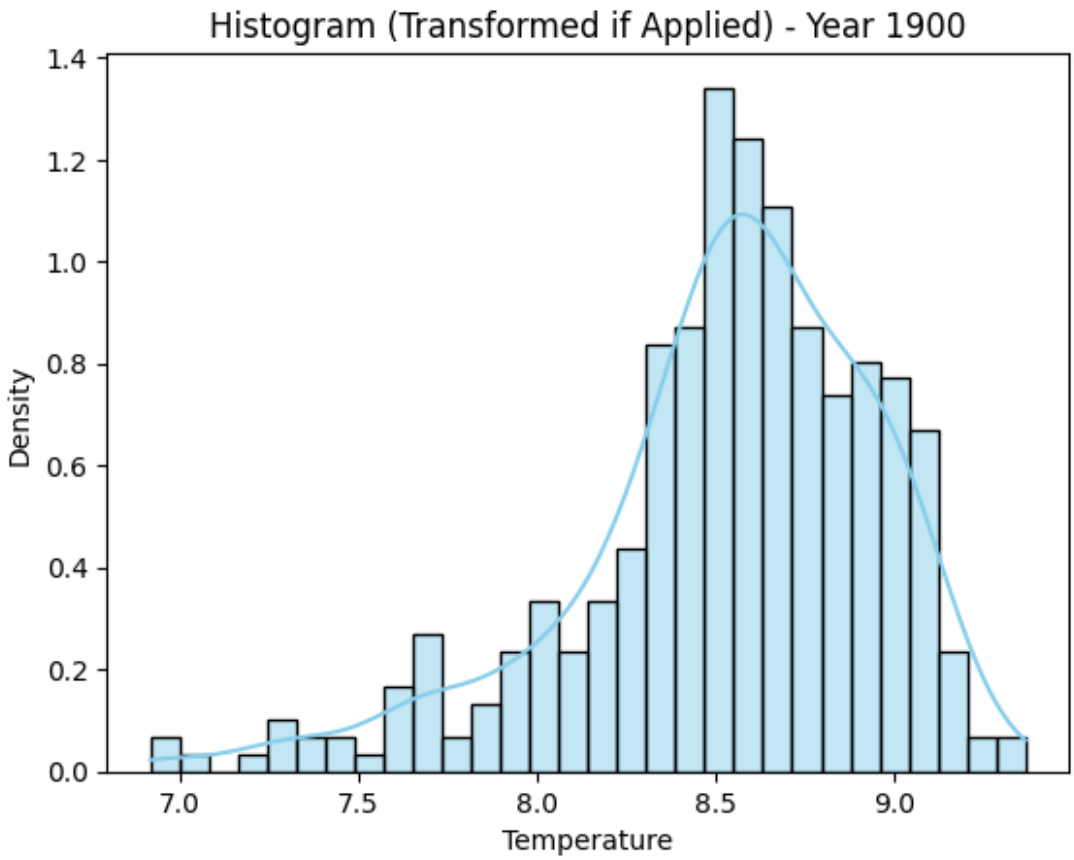
```
    plt.show()

    # Q-Q Plot to visually check normality
    stats.probplot(transformed_data, dist="norm", plot=plt)
    plt.title(f"Q-Q Plot (Transformed if Applied) - Year {year}")
    plt.show()
```

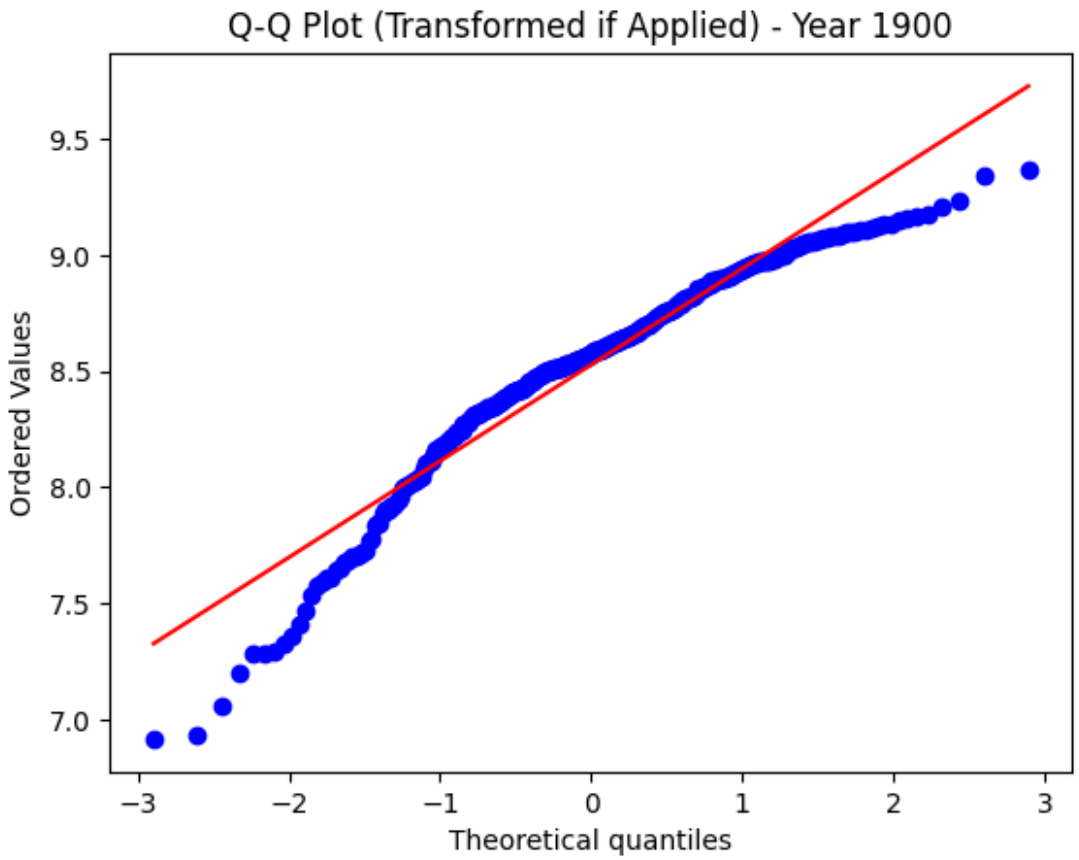
```
===== Year 1880 =====
Shapiro-Wilk test statistic: 0.9572, p-value: 0.0000
Data is NOT normally distributed. Applying Johnson transformation...

Post-transformation Shapiro-Wilk: stat = 0.9572, p = 0.0000
```



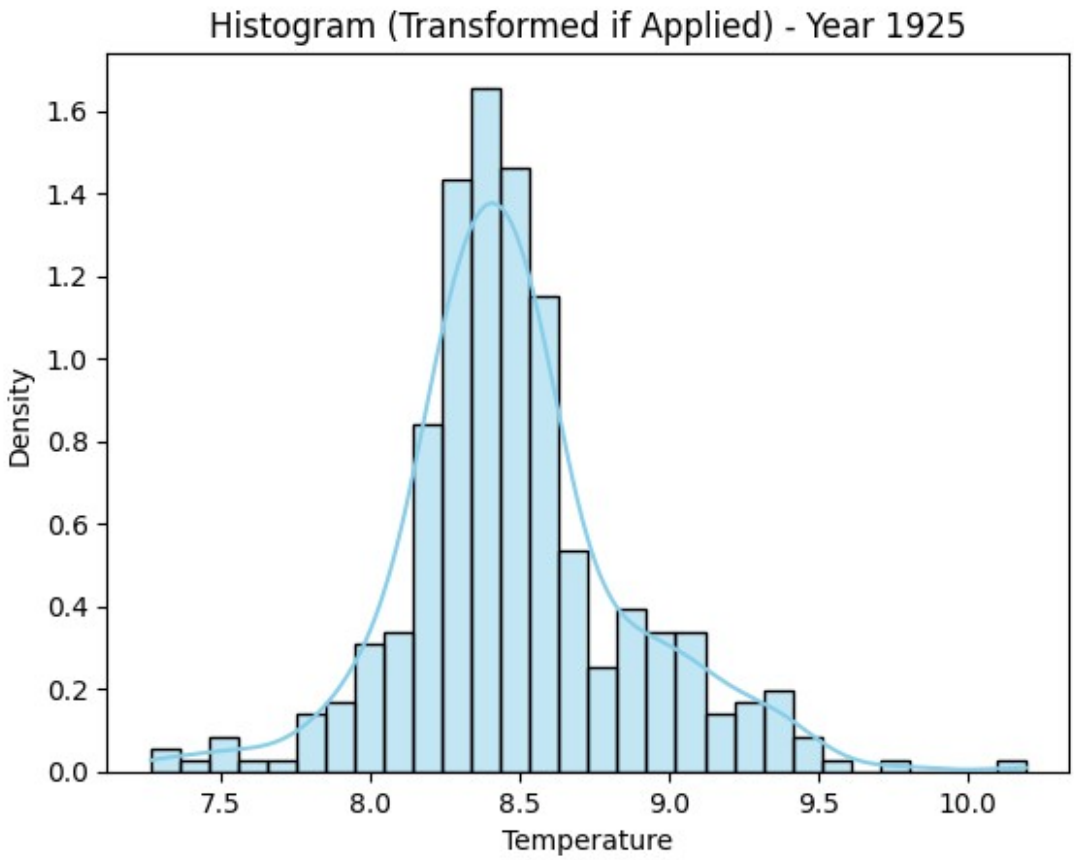


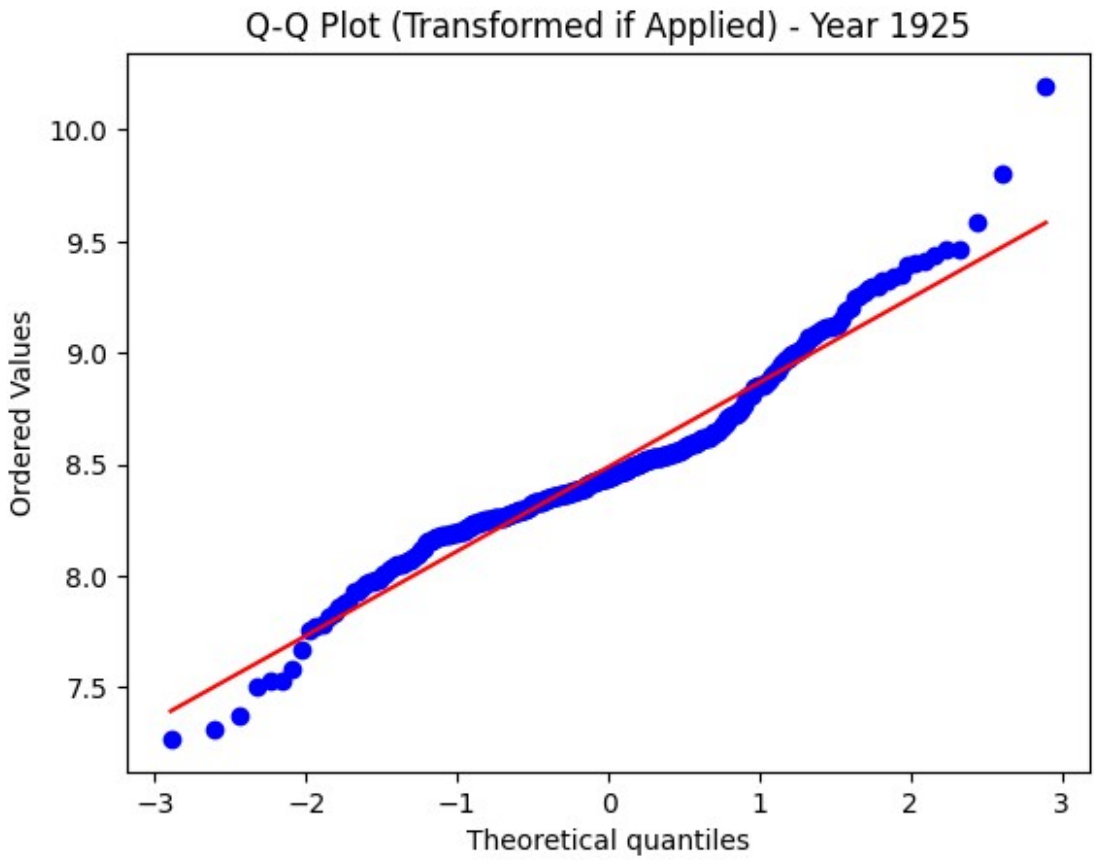

```
===== Year 1900 =====
Shapiro-Wilk test statistic: 0.9372, p-value: 0.0000
Data is NOT normally distributed. Applying Johnson transformation...

Post-transformation Shapiro-Wilk: stat = 0.9372, p = 0.0000
```


Histogram (Transformed if Applied) - Year 1900
Density
Temperature
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
7.0
7.5
8.0
8.5
9.0

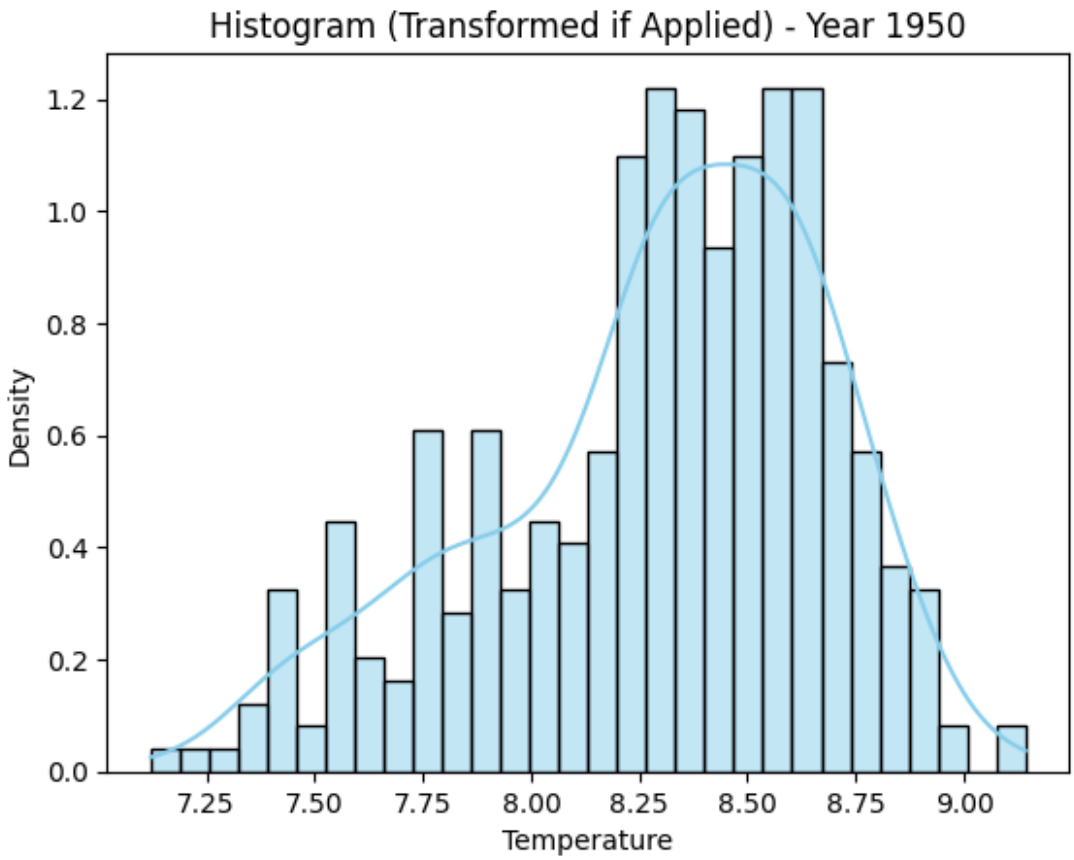


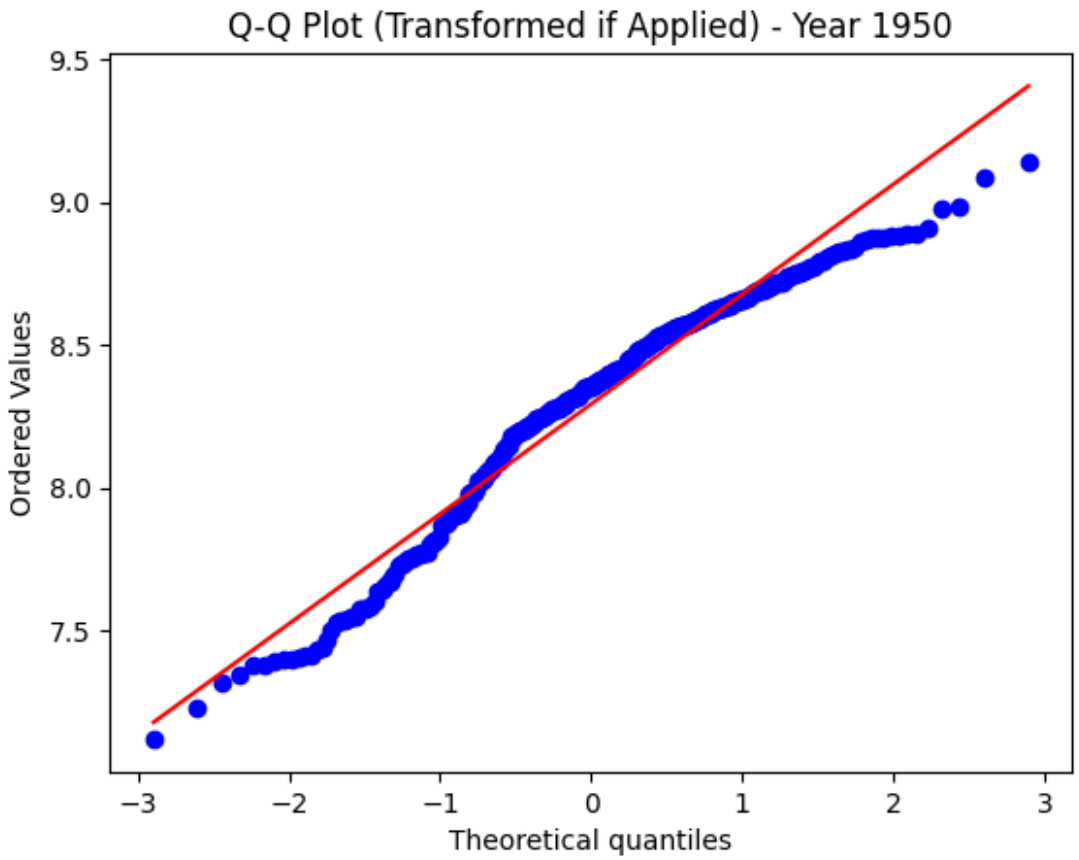

```
===== Year 1925 =====
Shapiro-Wilk test statistic: 0.9560, p-value: 0.0000
Data is NOT normally distributed. Applying Johnson transformation...

Post-transformation Shapiro-Wilk: stat = 0.9560, p = 0.0000
```


Histogram (Transformed if Applied) - Year 1925
Density
Temperature
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
7.5
8.0
8.5
9.0
9.5
10.0

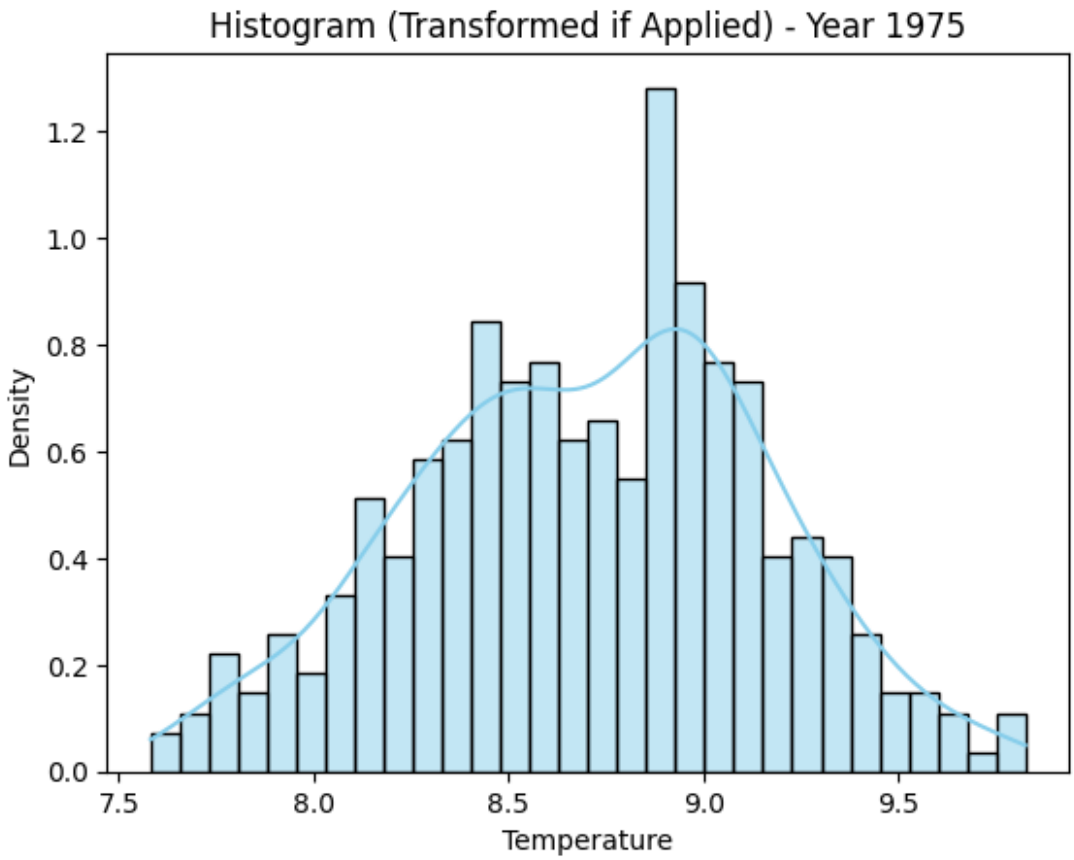


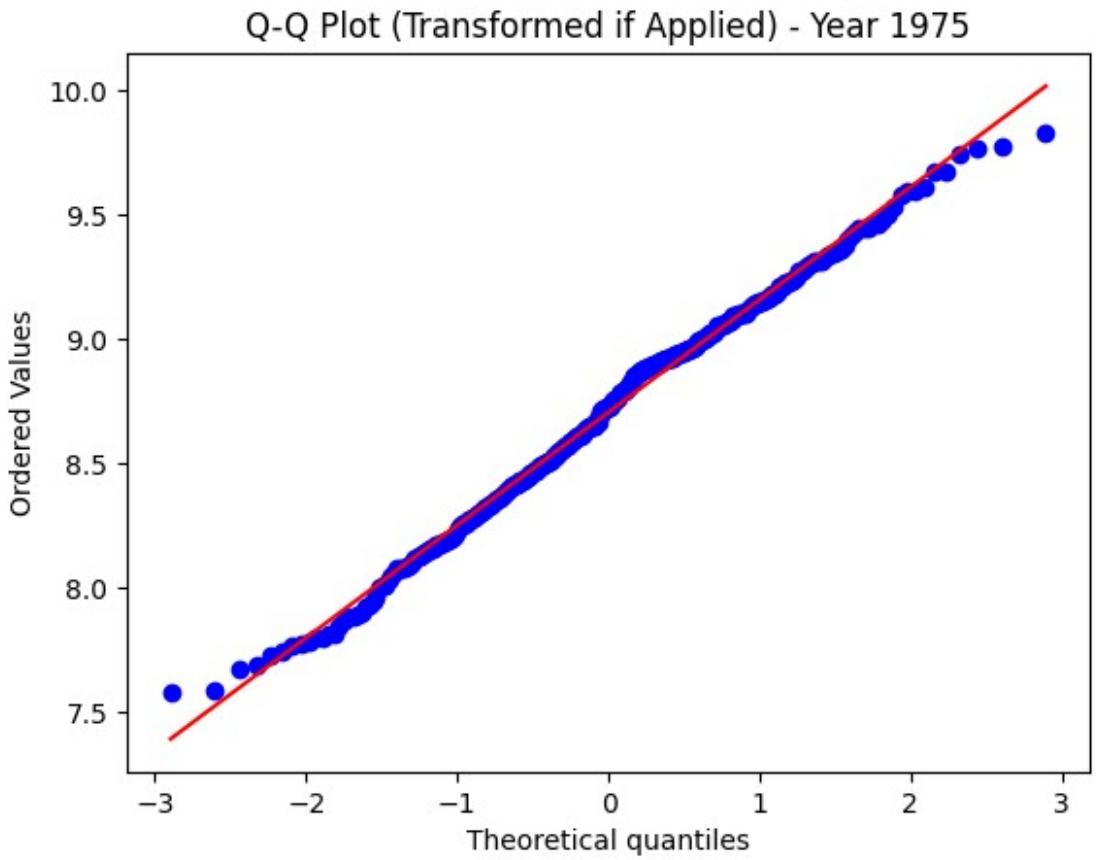

```
===== Year 1950 =====
Shapiro-Wilk test statistic: 0.9588, p-value: 0.0000
Data is NOT normally distributed. Applying Johnson transformation...

Post-transformation Shapiro-Wilk: stat = 0.9588, p = 0.0000
```


Histogram (Transformed if Applied) - Year 1950
Density
Temperature
0.0
0.2
0.4
0.6
0.8
1.0
1.2
7.25
7.50
7.75
8.00
8.25
8.50
8.75
9.00

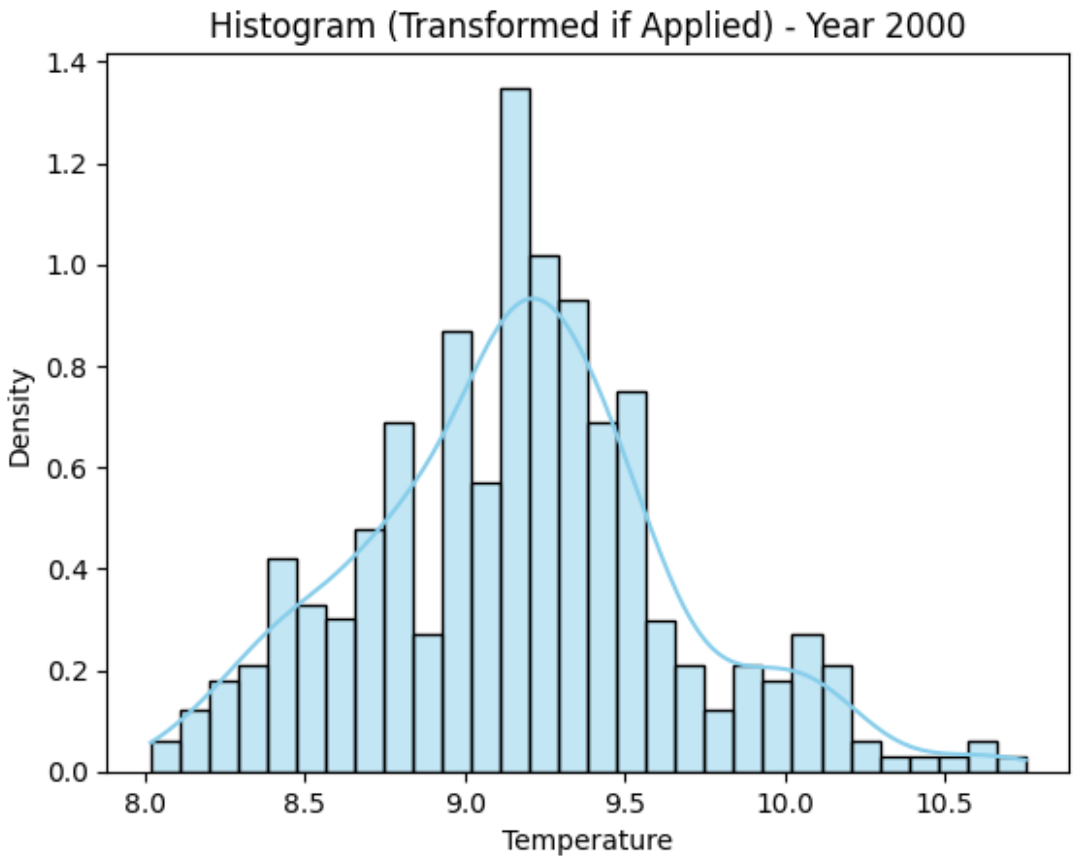


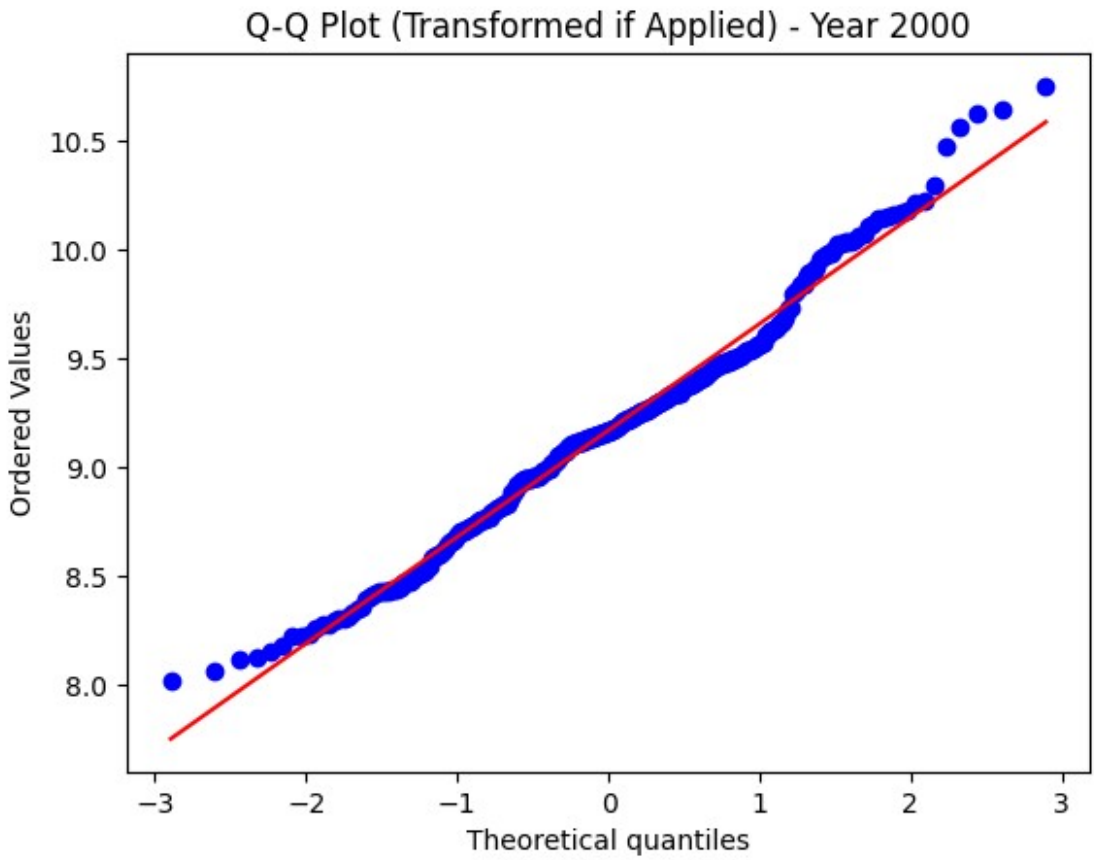

```
===== Year 1975 =====
Shapiro-Wilk test statistic: 0.9933, p-value: 0.1057
Data appears to be normally distributed.
```


Histogram (Transformed if Applied) - Year 1975
Density
Temperature
0.0
0.2
0.4
0.6
0.8
1.0
1.2
7.5
8.0
8.5
9.0
9.5

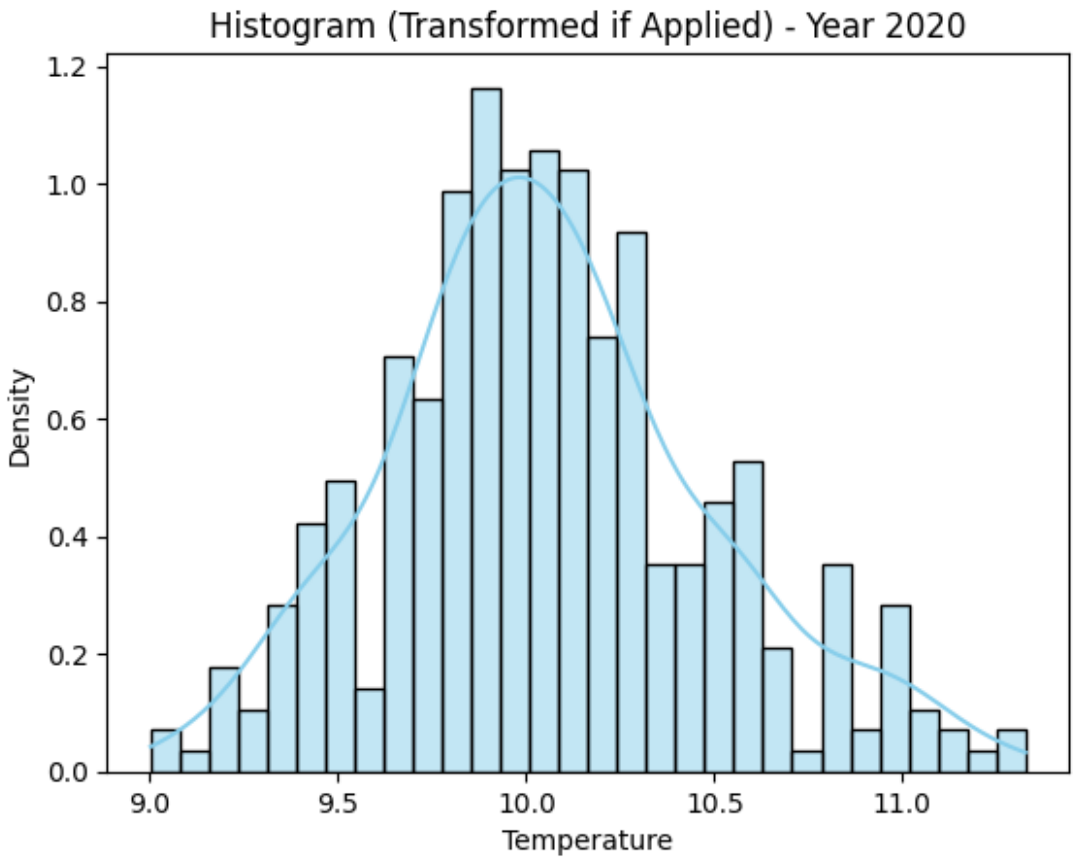


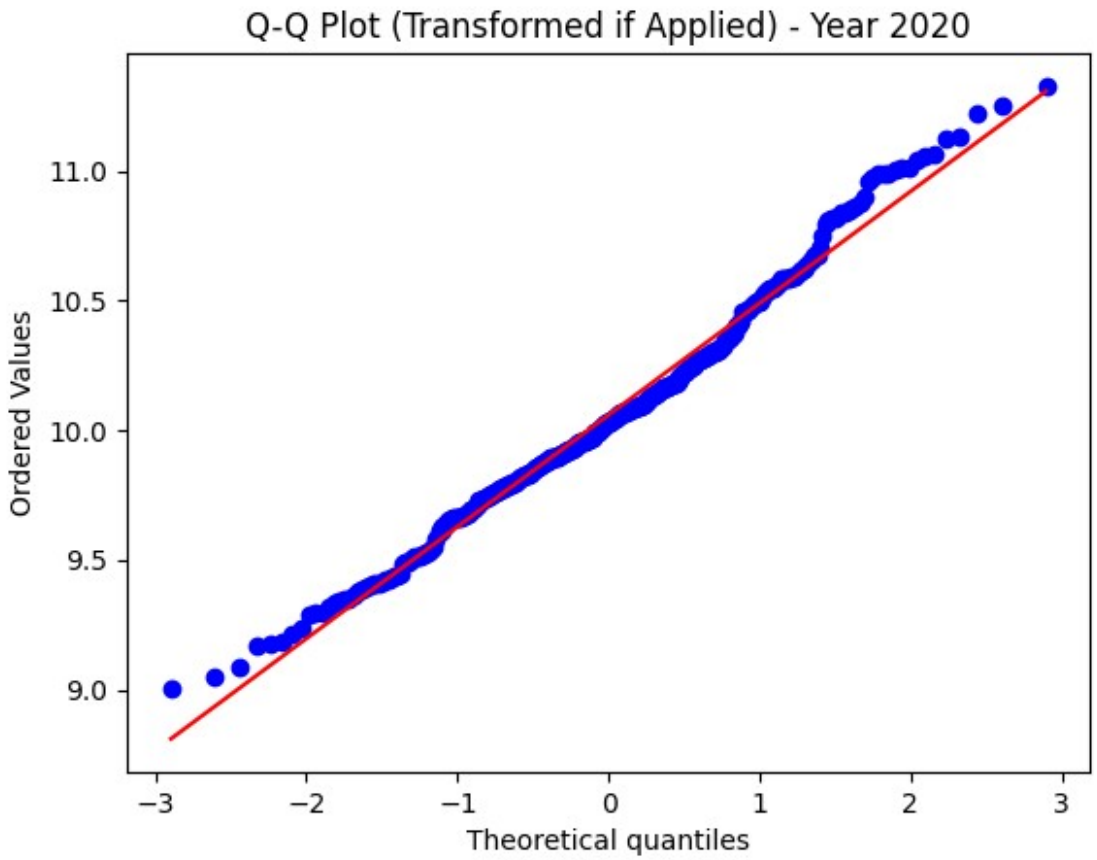

```
===== Year 2000 =====
Shapiro-Wilk test statistic: 0.9862, p-value: 0.0015
Data is NOT normally distributed. Applying Johnson transformation...

Post-transformation Shapiro-Wilk: stat = 0.9862, p = 0.0015
```


Histogram (Transformed if Applied) - Year 2000
Density
Temperature
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
8.0
8.5
9.0
9.5
10.0
10.5




```
===== Year 2020 =====
Shapiro-Wilk test statistic: 0.9865, p-value: 0.0017
Data is NOT normally distributed. Applying Johnson transformation...

Post-transformation Shapiro-Wilk: stat = 0.9865, p = 0.0017
```


Histogram (Transformed if Applied) - Year 2020
Density
Temperature
0.0
0.2
0.4
0.6
0.8
1.0
1.2
9.0
9.5
10.0
10.5
11.0




```
import pandas as pd
import numpy as np
from scipy import stats

def detect_outliers_std(df, column, threshold=2.5):
    """
    Detects outliers in a DataFrame column based on standard
deviations from the mean.

    Args:
        df (pd.DataFrame): The DataFrame to analyze.
        column (str): The name of the column to check for outliers.
        threshold (float): The number of standard deviations from the
mean to consider as an outlier.
                           Defaults to 2.5, as used in Hansen et al.
(2010).

    Returns:
        pd.DataFrame: A DataFrame containing the outliers and their
values.
    """
    mean = df[column].mean()
    std = df[column].std()
```

```
    outliers = df[(df[column] > mean + threshold * std) | (df[column]
< mean - threshold * std)]
    return outliers

def detect_inconsistent_neighbors(df, column='Temperature',
year_col='Year', threshold=2.5):
    """
    Detects data points that are inconsistent with neighboring years
based on standard deviations.
    This is a simplified version of the neighbor comparison mentioned
in Hansen et al. (2010).

    Args:
        df (pd.DataFrame): The DataFrame to analyze, containing 'Year'
and the temperature column.
        column (str): The name of the column to check for
inconsistencies (e.g., 'Temperature').
        year_col (str): The name of the year column.
        threshold (float): Threshold for considering a point as
inconsistent.

    Returns:
        pd.DataFrame: DataFrame of inconsistent data points.
    """

    df_shifted = df.copy()
    df_shifted['Temp_lag1'] = df_shifted[column].shift(1)
    df_shifted['Temp_lead1'] = df_shifted[column].shift(-1)

    df_shifted['Mean_Neighbor'] = df_shifted[['Temp_lag1',
'Temp_lead1']].mean(axis=1)
    df_shifted['Std_Neighbor'] = df_shifted[['Temp_lag1',
'Temp_lead1']].std(axis=1)

    df_shifted['Outlier_High'] = (df_shifted[column] >
df_shifted['Mean_Neighbor'] + threshold * df_shifted['Std_Neighbor'])
    df_shifted['Outlier_Low'] = (df_shifted[column] <
df_shifted['Mean_Neighbor'] - threshold * df_shifted['Std_Neighbor'])

    inconsistent_points = df_shifted[(df_shifted['Outlier_High'] ==
True) | (df_shifted['Outlier_Low'] == True)]

    return inconsistent_points[['Year', column]]  # Return only Year
and Temperature



# Example Usage:
# Overall outlier detection
overall_temp_outliers = detect_outliers_std(df1, 'Temperature')
```

```
print("Overall Temperature Outliers (Std Dev):\n",
overall_temp_outliers)

# Outlier detection by year (e.g., for each year's temperature)
yearly_outliers = df1.groupby('Year').apply(detect_outliers_std,
column='Temperature')
print("\nYearly Temperature Outliers (Std Dev):\n", yearly_outliers)

# Detect points inconsistent with neighboring years
inconsistent_neighbor_points = detect_inconsistent_neighbors(df1)
print("\nInconsistent Neighboring Year Points:\n",
inconsistent_neighbor_points)
```

```
Overall Temperature Outliers (Std Dev):
        Year  Month  Temperature        Year Group         Half Year
Group 10
72     1880      3        6.968               NaN   First Half
1880-1889
73     1880      3        6.945               NaN   First Half
1880-1889
1063   1882     11        6.950  (1880.0, 1900.0]   First Half
1880-1889
1064   1882     11        6.921  (1880.0, 1900.0]   First Half
1880-1889
1070   1882     12        6.856  (1880.0, 1900.0]   First Half
1880-1889
...     ...    ...          ...               ...          ...
...
51964  2022      4       10.816               NaN  Second Half
NaN
51965  2022      4       10.653               NaN  Second Half
NaN
52025  2022      6       10.385               NaN  Second Half
NaN
52030  2022      6       10.394               NaN  Second Half
NaN
52031  2022      6       10.518               NaN  Second Half
NaN

[885 rows x 6 columns]

Yearly Temperature Outliers (Std Dev):
             Year  Month  Temperature        Year Group
Half  \
Year
1880 72     1880      3        6.968               NaN   First Half
     73     1880      3        6.945               NaN   First Half
1881 410    1881      2        7.413  (1880.0, 1900.0]   First Half
     687    1881     11        7.447  (1880.0, 1900.0]   First Half
     688    1881     11        7.280  (1880.0, 1900.0]   First Half
```

```
...          ...    ...          ...                ...          ...
2021 51557  2021      2        8.957               NaN  Second Half
2022 51960  2022      4       11.137               NaN  Second Half
     51961  2022      4       11.352               NaN  Second Half
     51962  2022      4       11.291               NaN  Second Half
     51963  2022      4       11.014               NaN  Second Half

           Year Group 10
Year
1880 72        1880-1889
     73        1880-1889
1881 410       1880-1889
     687       1880-1889
     688       1880-1889
...                  ...
2021 51557           NaN
2022 51960           NaN
     51961           NaN
     51962           NaN
     51963           NaN

[942 rows x 6 columns]

Inconsistent Neighboring Year Points:
        Year  Temperature
1      1880        7.998
2      1880        7.917
3      1880        7.975
5      1880        7.847
6      1880        7.944
...     ...          ...
52035  2022       10.074
52045  2022        9.348
52052  2022        9.786
52058  2022        9.376
52072  2022       10.229

[6943 rows x 2 columns]
```

Missing Values: Check for missing or inconsistent records.

Outlier Detection: Boxplots and IQR method.

Trend Analysis: Moving averages, seasonal decomposition.

Visualization:

1.Time series plots for different periods.

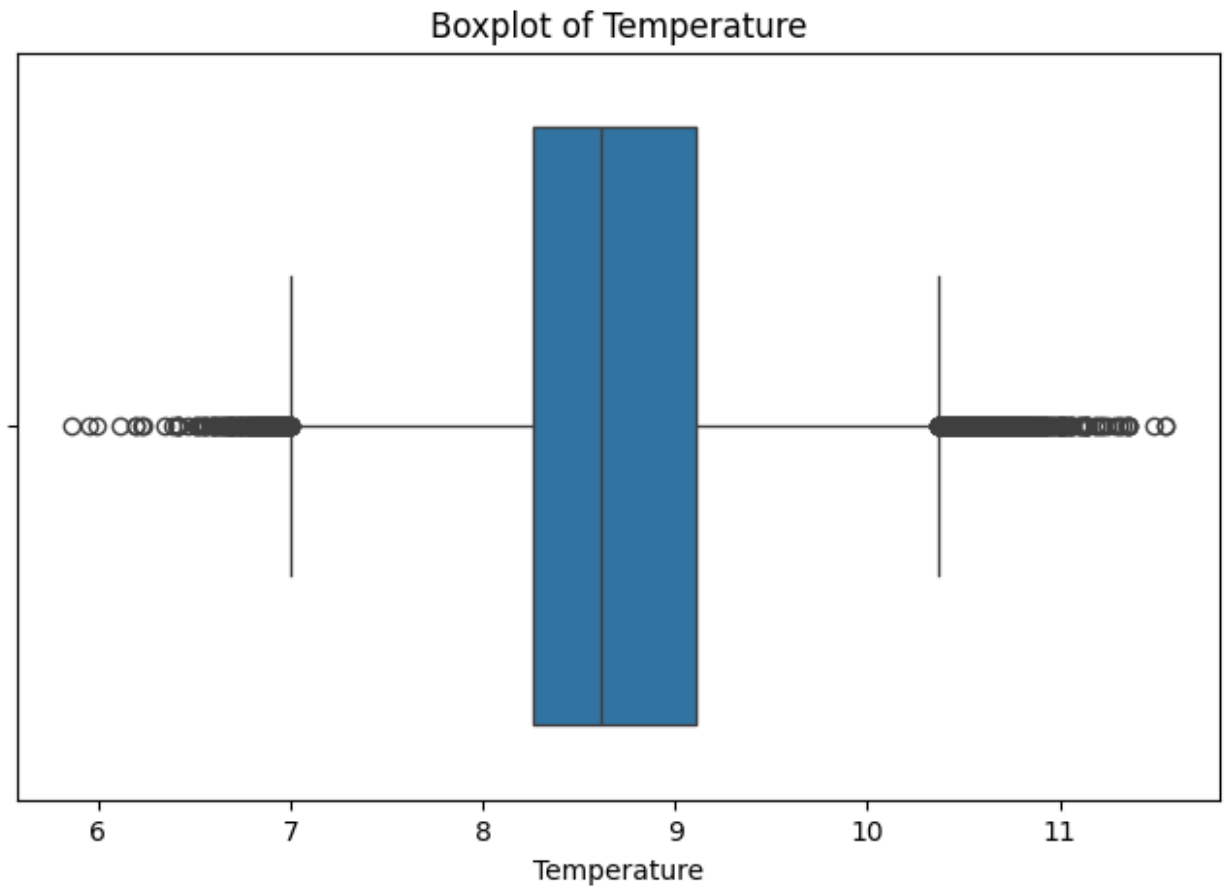
2.Rolling mean and standard deviation.

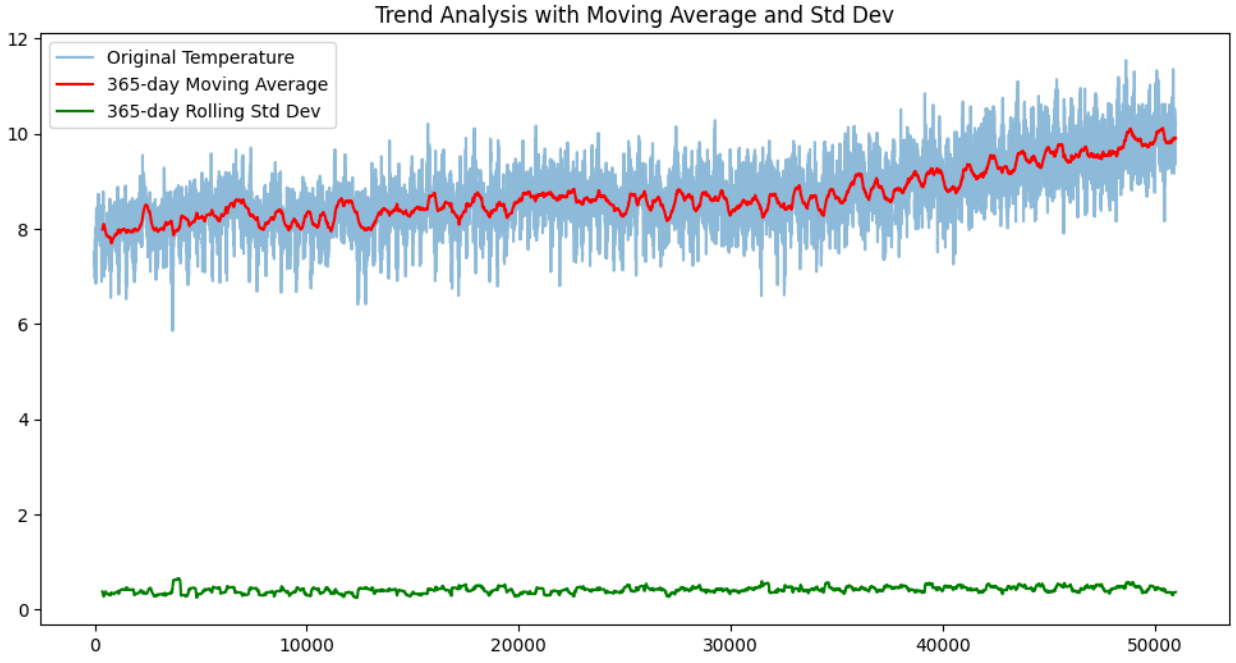
3. Histogram of temperature anomalies.

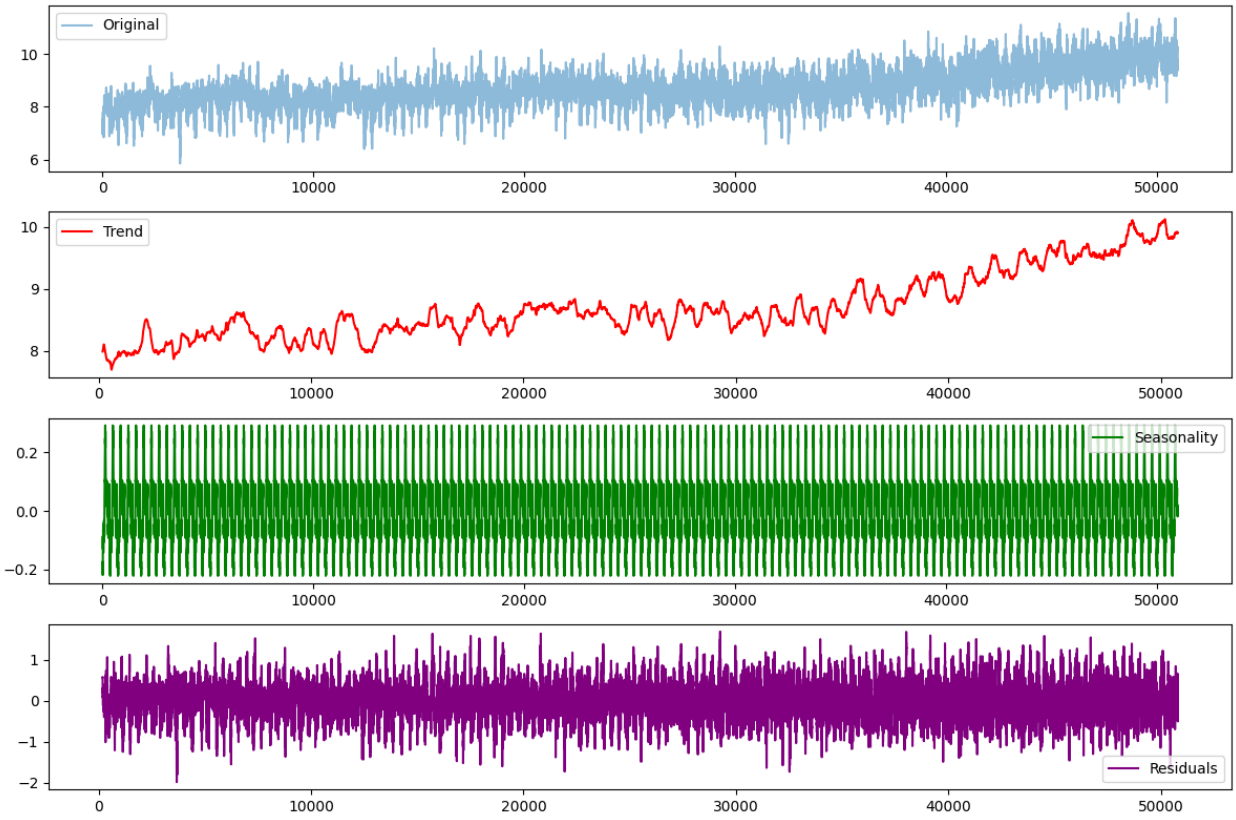
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose



# Display basic info
print(df.info())
print(df.head())

# Check for missing values
missing_values = df.isnull().sum()
print("Missing Values:\n", missing_values)

# Handling missing values (if any)
df.dropna(inplace=True)  # Remove missing records
df.reset_index(drop=True, inplace=True)

# Outlier Detection using IQR
Q1 = df['Temperature'].quantile(0.25)
Q3 = df['Temperature'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers = df[(df['Temperature'] < lower_bound) | (df['Temperature'] >
upper_bound)]
print(f"Number of Outliers: {len(outliers)}")

# Boxplot for outlier visualization
plt.figure(figsize=(8, 5))
```

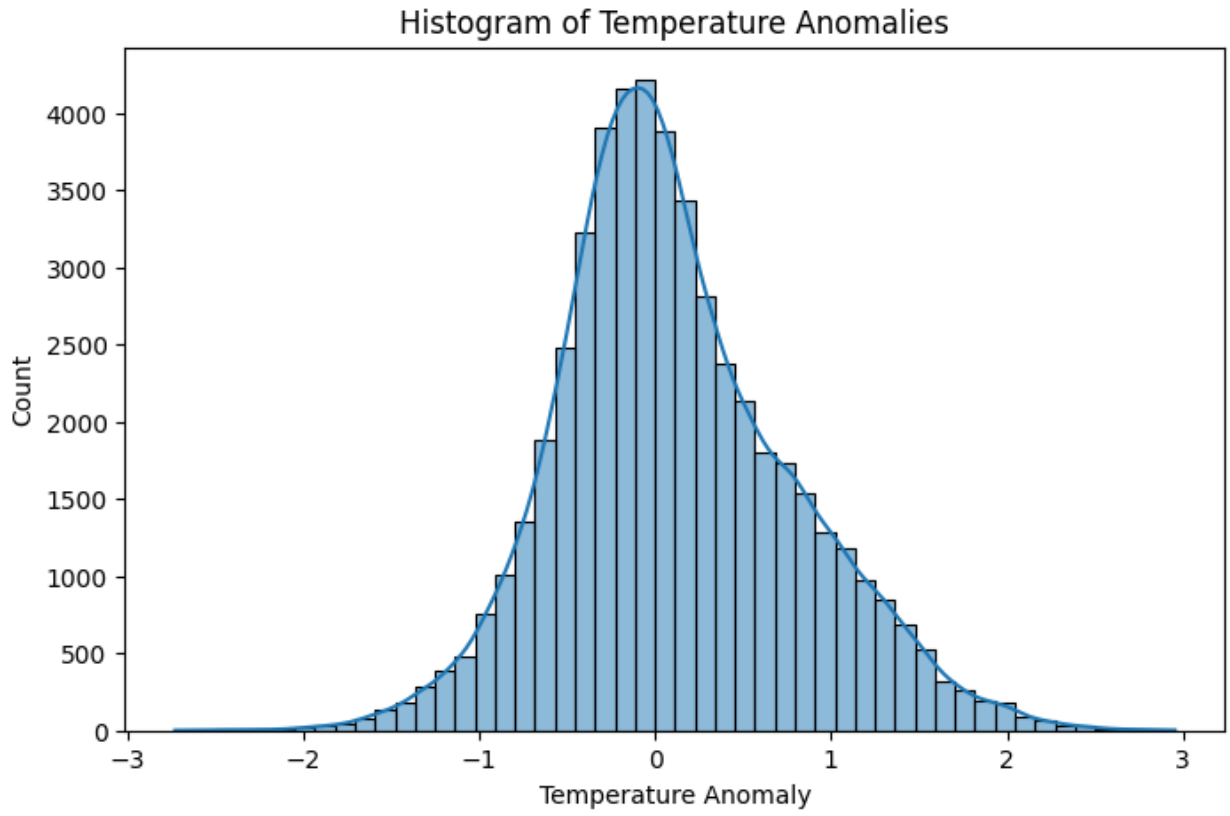
```
sns.boxplot(x=df['Temperature'])
plt.title("Boxplot of Temperature")
plt.show()

# Trend Analysis using Moving Average
df['Rolling_Mean'] = df['Temperature'].rolling(window=365).mean()
df['Rolling_Std'] = df['Temperature'].rolling(window=365).std()

plt.figure(figsize=(12, 6))
plt.plot(df['Temperature'], label="Original Temperature", alpha=0.5)
plt.plot(df['Rolling_Mean'], label="365-day Moving Average",
color='red')
plt.plot(df['Rolling_Std'], label="365-day Rolling Std Dev",
color='green')
plt.title("Trend Analysis with Moving Average and Std Dev")
plt.legend()
plt.show()

# Seasonal Decomposition
decomposition = seasonal_decompose(df['Temperature'], period=365,
model='additive')

plt.figure(figsize=(12, 8))
plt.subplot(411)
plt.plot(df['Temperature'], label="Original", alpha=0.5)
plt.legend()

plt.subplot(412)
plt.plot(decomposition.trend, label="Trend", color='red')
plt.legend()

plt.subplot(413)
plt.plot(decomposition.seasonal, label="Seasonality", color='green')
plt.legend()

plt.subplot(414)
plt.plot(decomposition.resid, label="Residuals", color='purple')
plt.legend()

plt.tight_layout()
plt.show()

# Histogram of temperature anomalies
plt.figure(figsize=(8, 5))
sns.histplot(df['Anomaly'], bins=50, kde=True)
plt.title("Histogram of Temperature Anomalies")
plt.xlabel("Temperature Anomaly")
plt.show()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51349 entries, 0 to 51348
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Year          51349 non-null  int64  
 1   Month         51349 non-null  int64  
 2   Day           51349 non-null  int64  
 3   Day of Year   51349 non-null  int64  
 4   Anomaly       51349 non-null  float64
 5   Temperature   51349 non-null  float64
 6   Rolling_Mean  50985 non-null  float64
 7   Rolling_Std   50985 non-null  float64
dtypes: float64(4), int64(4)
memory usage: 3.1 MB
None
   Year  Month  Day  Day of Year  Anomaly  Temperature  
Rolling_Mean  \
0  1881     12   29          363    0.232        8.822           NaN

1  1881     12   30          364   -0.015        8.575           NaN

2  1881     12   31          365    0.013        8.603           NaN

3  1882      1    1            1    0.020        8.610           NaN

4  1882      1    2            2   -0.043        8.547           NaN


   Rolling_Std
0          NaN
1          NaN
2          NaN
3          NaN
4          NaN
Missing Values:
 Year              0
Month             0
Day               0
Day of Year       0
Anomaly           0
Temperature       0
Rolling_Mean    364
Rolling_Std     364
dtype: int64
Number of Outliers: 881
```


Boxplot of Temperature
6
7
8
9
10
11
Temperature


Trend Analysis with Moving Average and Std Dev
Original Temperature
365-day Moving Average
365-day Rolling Std Dev
12
10
8
6
4
2
0
0
10000
20000
30000
40000
50000

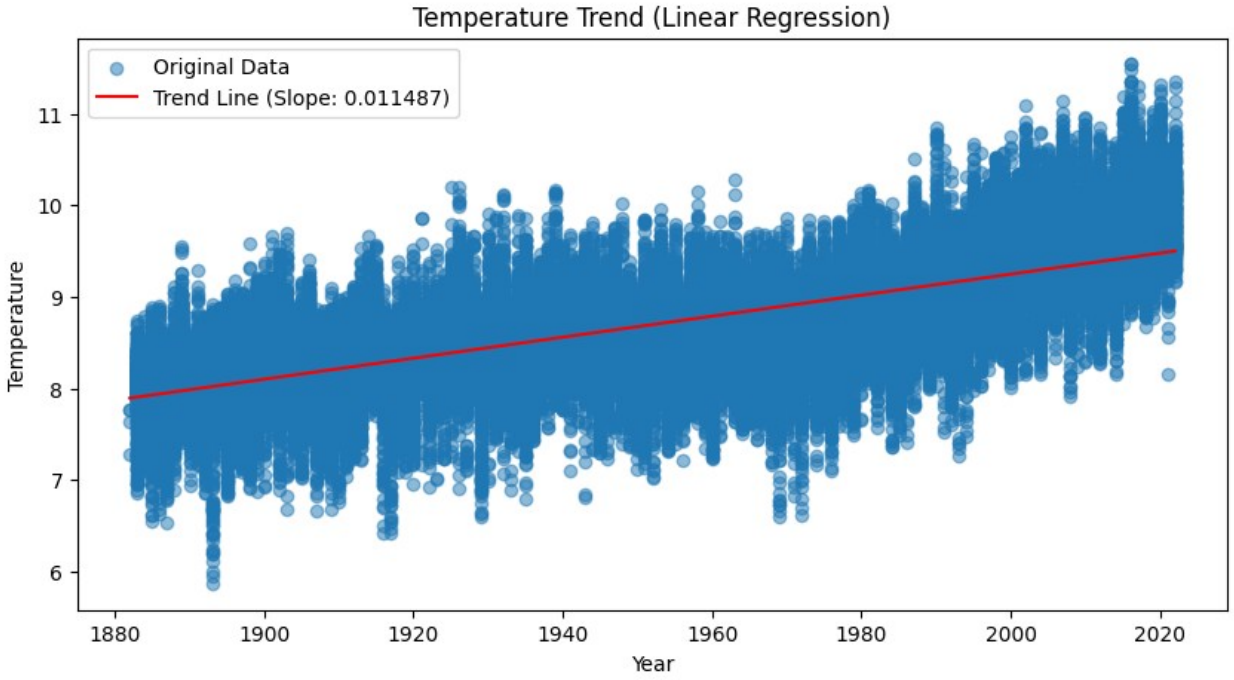

Original
Trend
Seasonality
Residuals

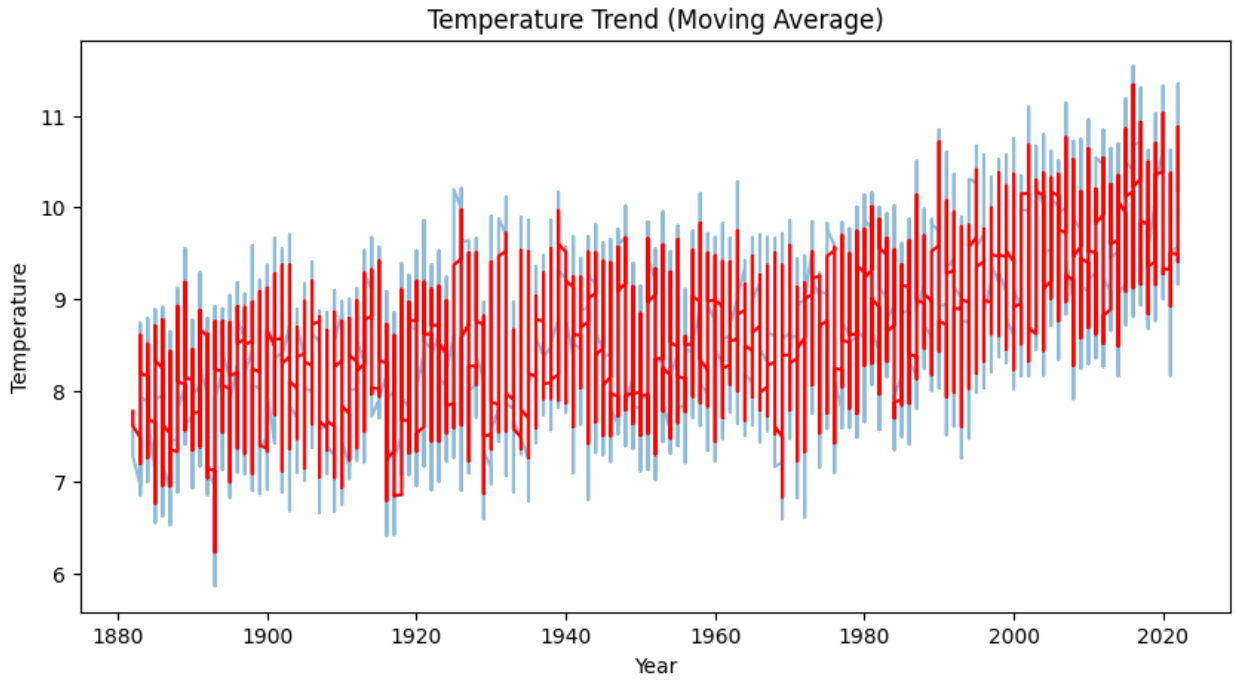

Histogram of Temperature Anomalies
Count
Temperature Anomaly

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import theilslopes
from sklearn.linear_model import LinearRegression
from statsmodels.tsa.stattools import adfuller
import pymannkendall as mk

X = df[['Year']].values.reshape(-1, 1)
y = df['Temperature'].values

model = LinearRegression()
model.fit(X, y)

trend_slope = model.coef_[0]
intercept = model.intercept_
print(f"Linear Regression Trend Slope: {trend_slope:.6f}")

# Plot the trend line
plt.figure(figsize=(10, 5))
plt.scatter(df['Year'], df['Temperature'], label="Original Data",
alpha=0.5)
plt.plot(df['Year'], model.predict(X), color='red', label=f"Trend Line
(Slope: {trend_slope:.6f})")
plt.xlabel("Year")
plt.ylabel("Temperature")
plt.title("Temperature Trend (Linear Regression)")
plt.legend()
plt.show()



df['Rolling_Mean'] = df['Temperature'].rolling(window=10,
min_periods=1).mean()

plt.figure(figsize=(10, 5))
plt.plot(df['Year'], df['Temperature'], label="Original Data",
alpha=0.5)
plt.plot(df['Year'], df['Rolling_Mean'], label="10-Year Moving
Average", color='red')
plt.xlabel("Year")
plt.ylabel("Temperature")
plt.title("Temperature Trend (Moving Average)")
plt.show()



adf_test = adfuller(df['Temperature'])
```

```python
print(f"ADF Statistic: {adf_test[0]}")
print(f"p-value: {adf_test[1]}")


methods = {
    "Linear Regression Slope": trend_slope,
    "ADF Test p-value": adf_test[1]
}

for method, value in methods.items():
    print(f"{method}: {value:.6f}")
```

```
Linear Regression Trend Slope: 0.011487
```

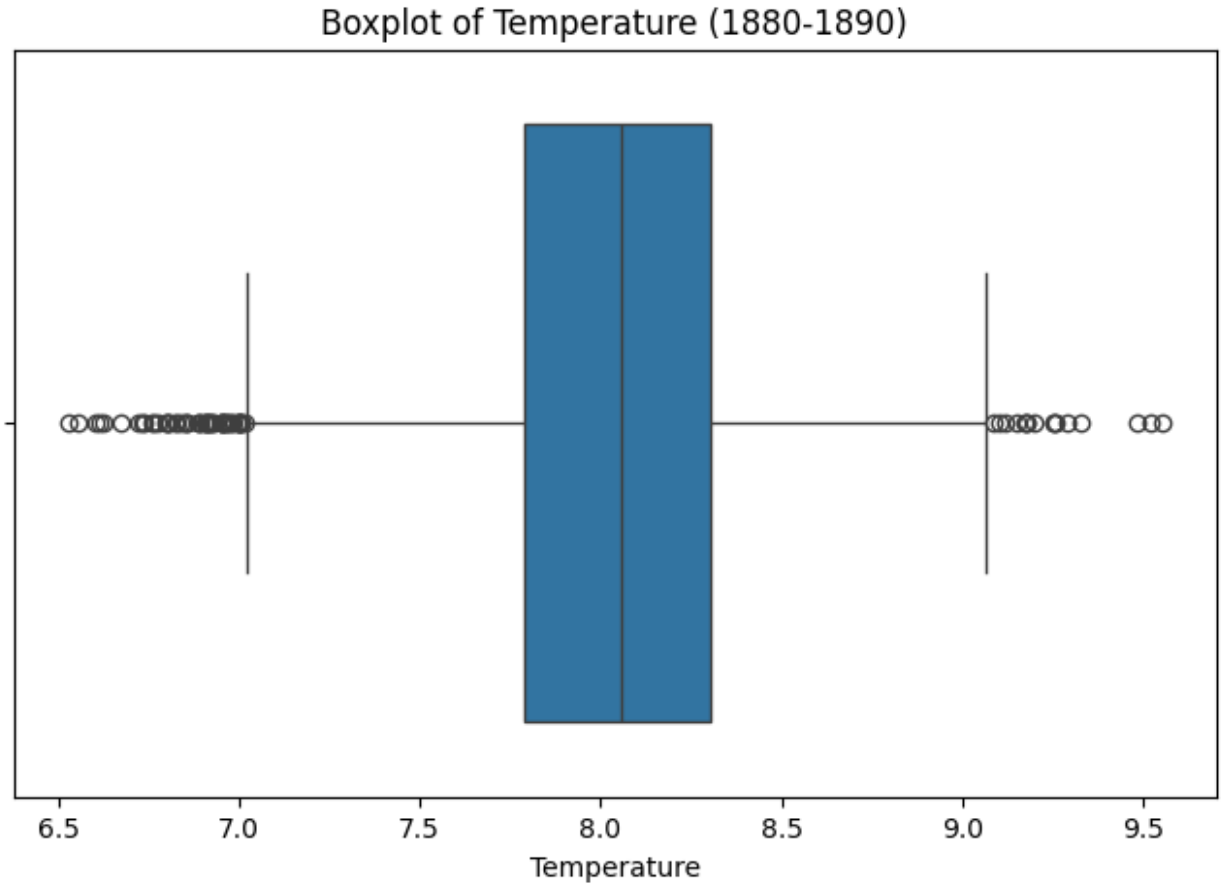


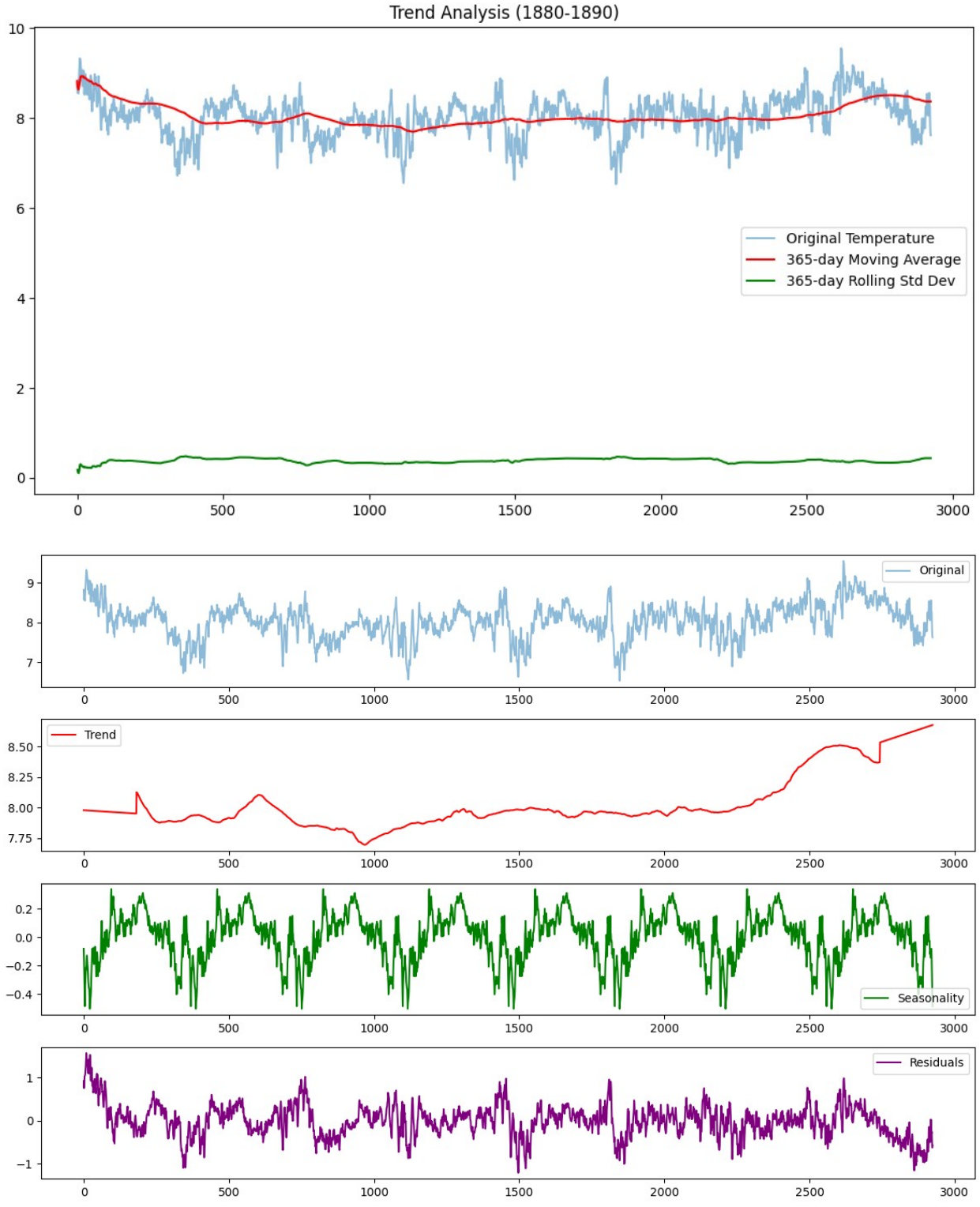


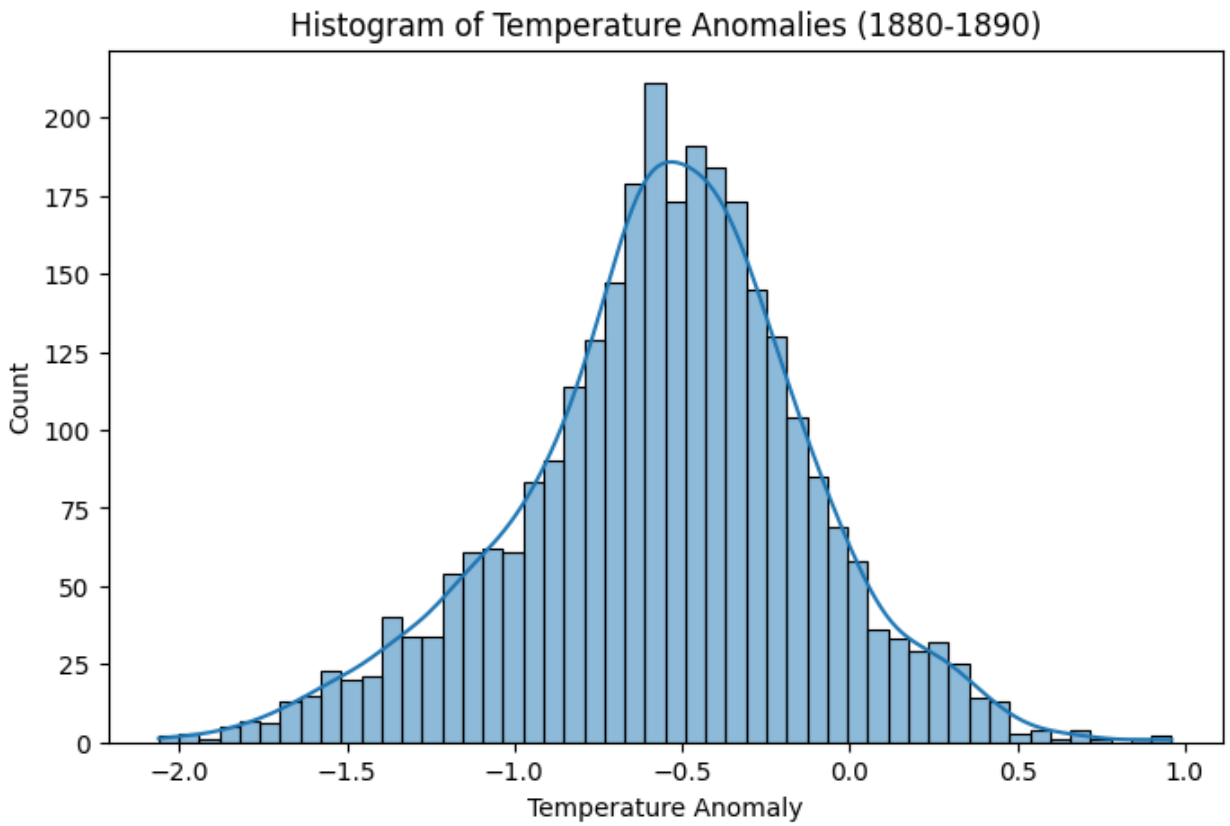

```
ADF Statistic: -9.437660805688283
p-value: 4.991882507815218e-16
Linear Regression Slope: 0.011487
ADF Test p-value: 0.000000

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose

# Assuming df contains the dataset with 'Year', 'Temperature', and
'Anomaly' columns
group_size = 10
start_year = 1880
end_year = 2020

for start in range(start_year, end_year, group_size):
    df_group = df[(df['Year'] >= start) & (df['Year'] < start +
group_size)].copy()
    print(f"Analyzing period: {start}-{start + group_size}")

    # Outlier Detection using IQR
    Q1 = df_group['Temperature'].quantile(0.25)
    Q3 = df_group['Temperature'].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df_group[(df_group['Temperature'] < lower_bound) |
```

```
(df_group['Temperature'] > upper_bound)]
    print(f"Number of Outliers: {len(outliers)}")

    # Boxplot for Outliers
    plt.figure(figsize=(8, 5))
    sns.boxplot(x=df_group['Temperature'])
    plt.title(f"Boxplot of Temperature ({start}-{start +
group_size})")
    plt.show()

    # Trend Analysis using Moving Average
    df_group['Rolling_Mean'] =
df_group['Temperature'].rolling(window=365, min_periods=1).mean()
    df_group['Rolling_Std'] =
df_group['Temperature'].rolling(window=365, min_periods=1).std()

    plt.figure(figsize=(12, 6))
    plt.plot(df_group['Temperature'], label="Original Temperature",
alpha=0.5)
    plt.plot(df_group['Rolling_Mean'], label="365-day Moving Average",
color='red')
    plt.plot(df_group['Rolling_Std'], label="365-day Rolling Std Dev",
color='green')
    plt.title(f"Trend Analysis ({start}-{start + group_size})")
    plt.legend()
    plt.show()

    # Seasonal Decomposition
    decomposition = seasonal_decompose(df_group['Temperature'],
period=365, model='additive', extrapolate_trend='freq')

    plt.figure(figsize=(12, 8))
    plt.subplot(411)
    plt.plot(df_group['Temperature'], label="Original", alpha=0.5)
    plt.legend()

    plt.subplot(412)
    plt.plot(decomposition.trend, label="Trend", color='red')
    plt.legend()

    plt.subplot(413)
    plt.plot(decomposition.seasonal, label="Seasonality",
color='green')
    plt.legend()

    plt.subplot(414)
    plt.plot(decomposition.resid, label="Residuals", color='purple')
    plt.legend()
```

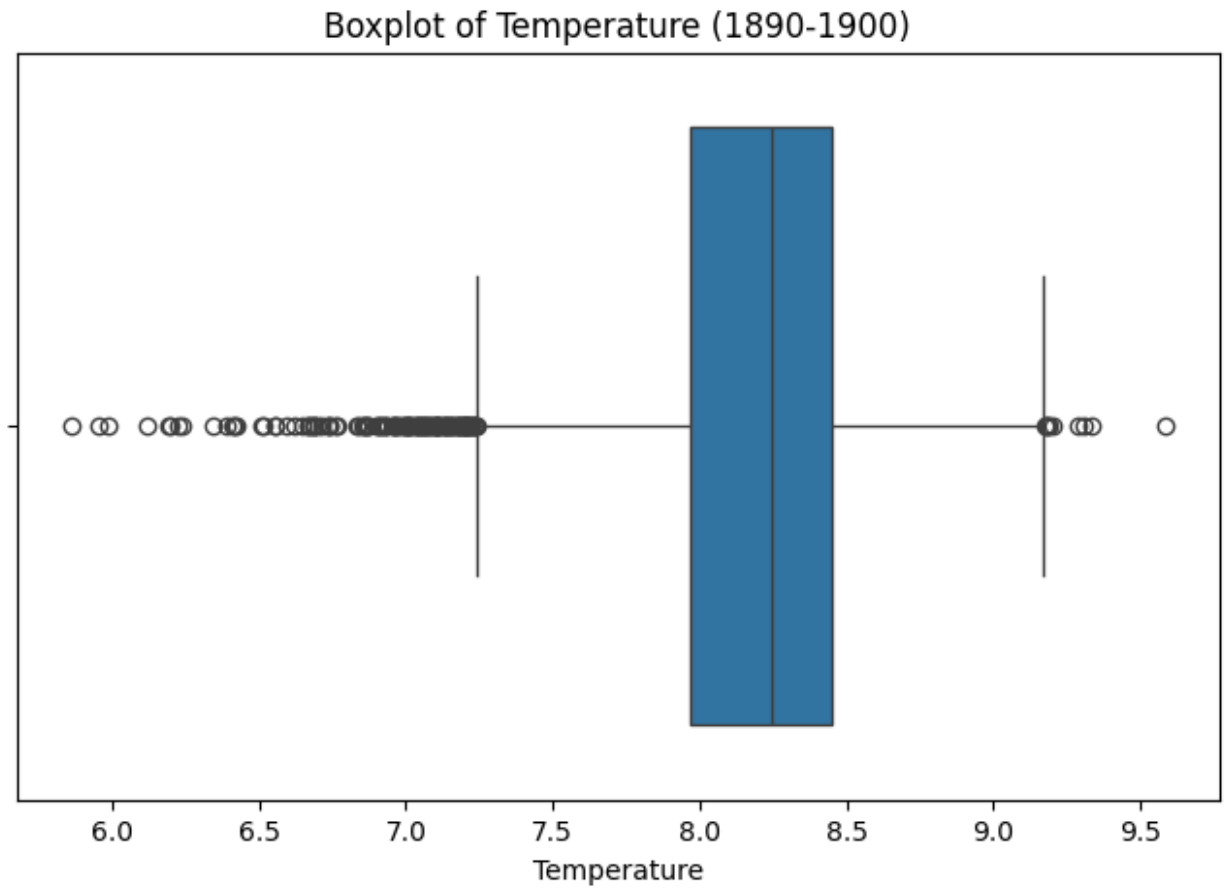
```
    plt.tight_layout()
    plt.show()

    # Histogram of Temperature Anomalies
    plt.figure(figsize=(8, 5))
    sns.histplot(df_group['Anomaly'], bins=50, kde=True)
    plt.title(f"Histogram of Temperature Anomalies ({start}-{start +
group_size})")
    plt.xlabel("Temperature Anomaly")
    plt.show()

Analyzing period: 1880-1890
Number of Outliers: 69
```

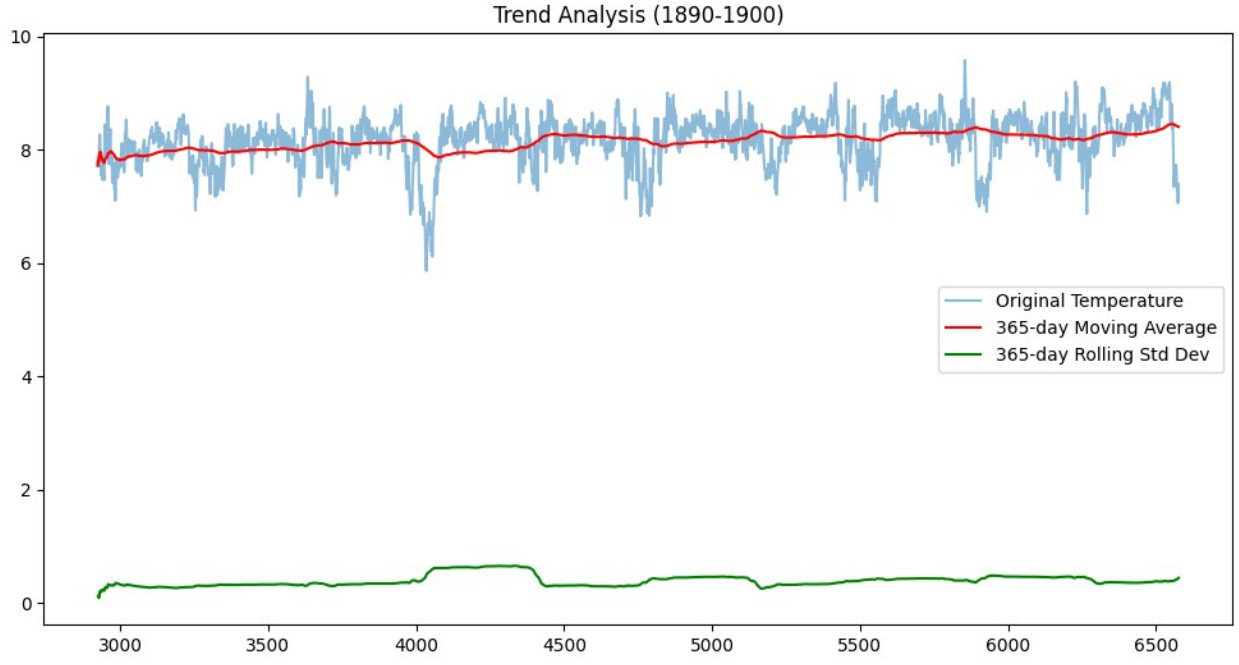


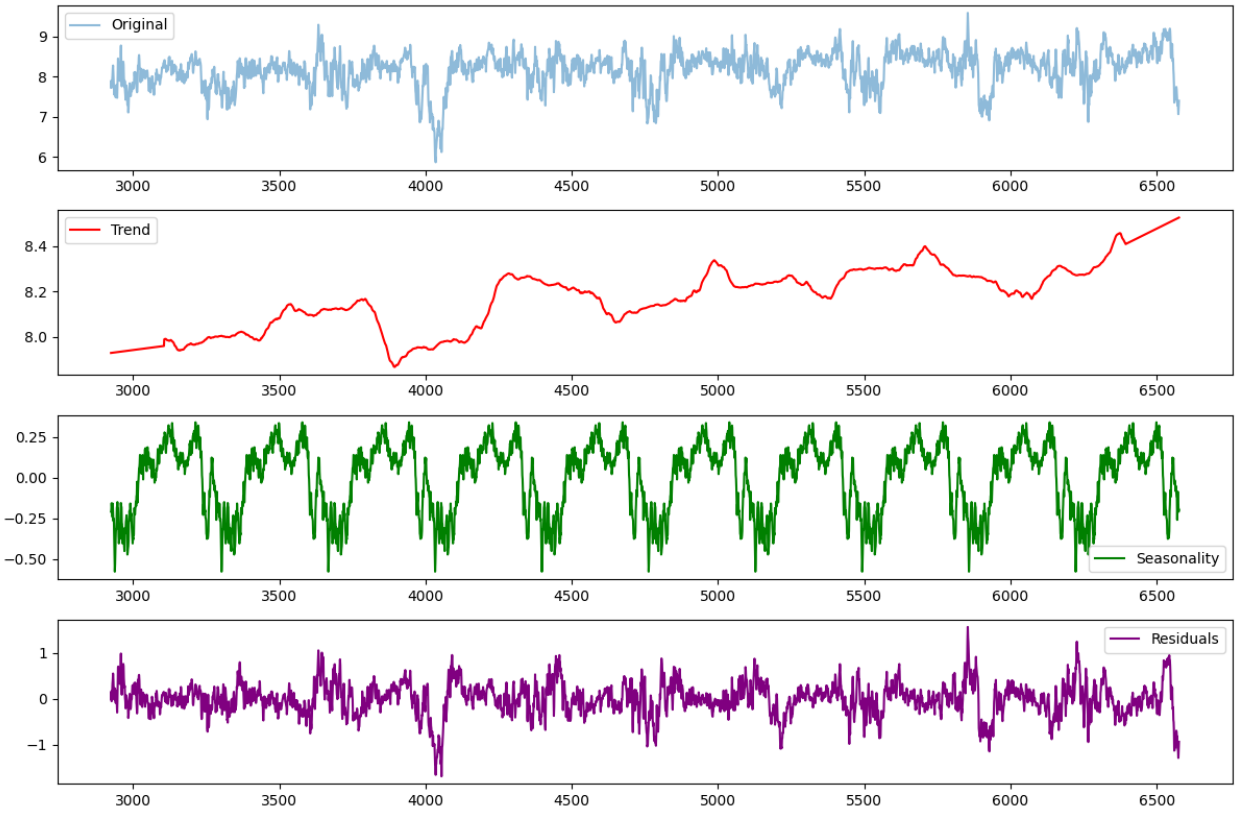

Trend Analysis (1880-1890)
Original Temperature
365-day Moving Average
365-day Rolling Std Dev
Original
Trend
Seasonality
Residuals

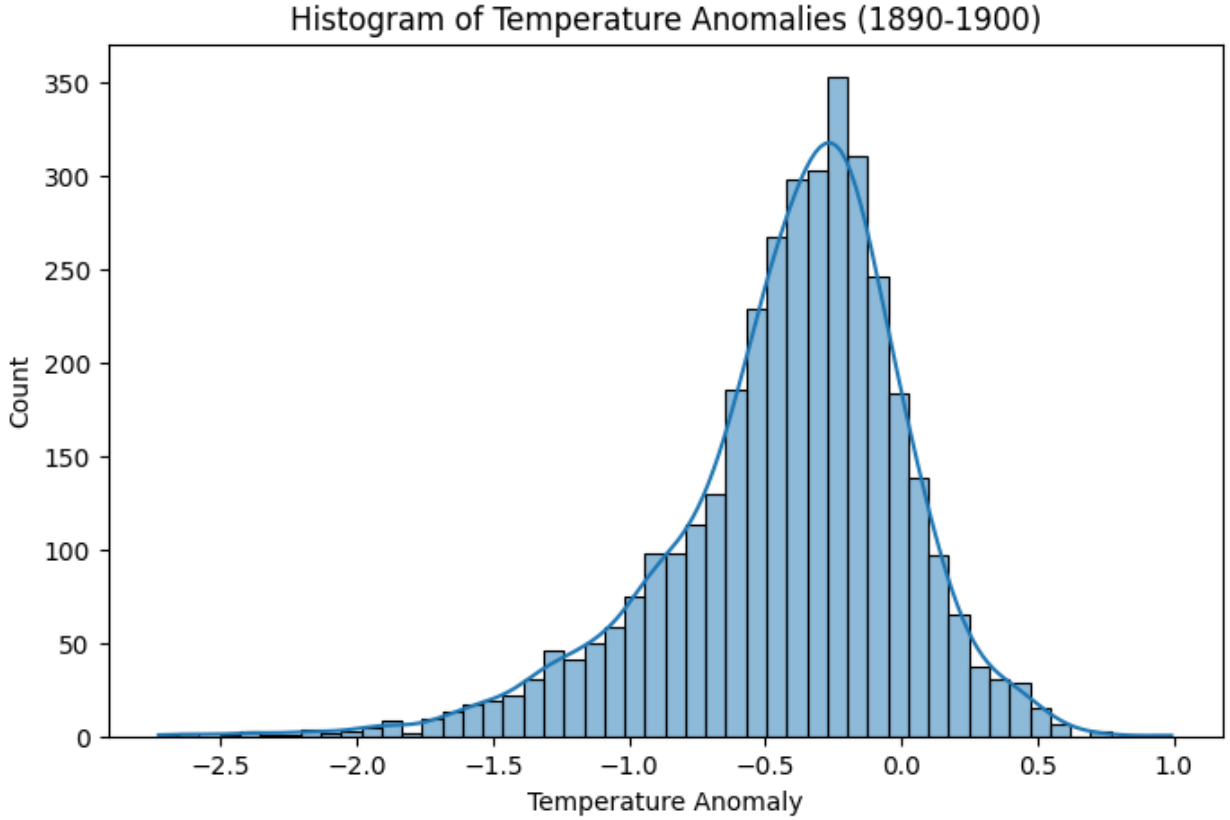



```
Analyzing period: 1890-1900
Number of Outliers: 137
```

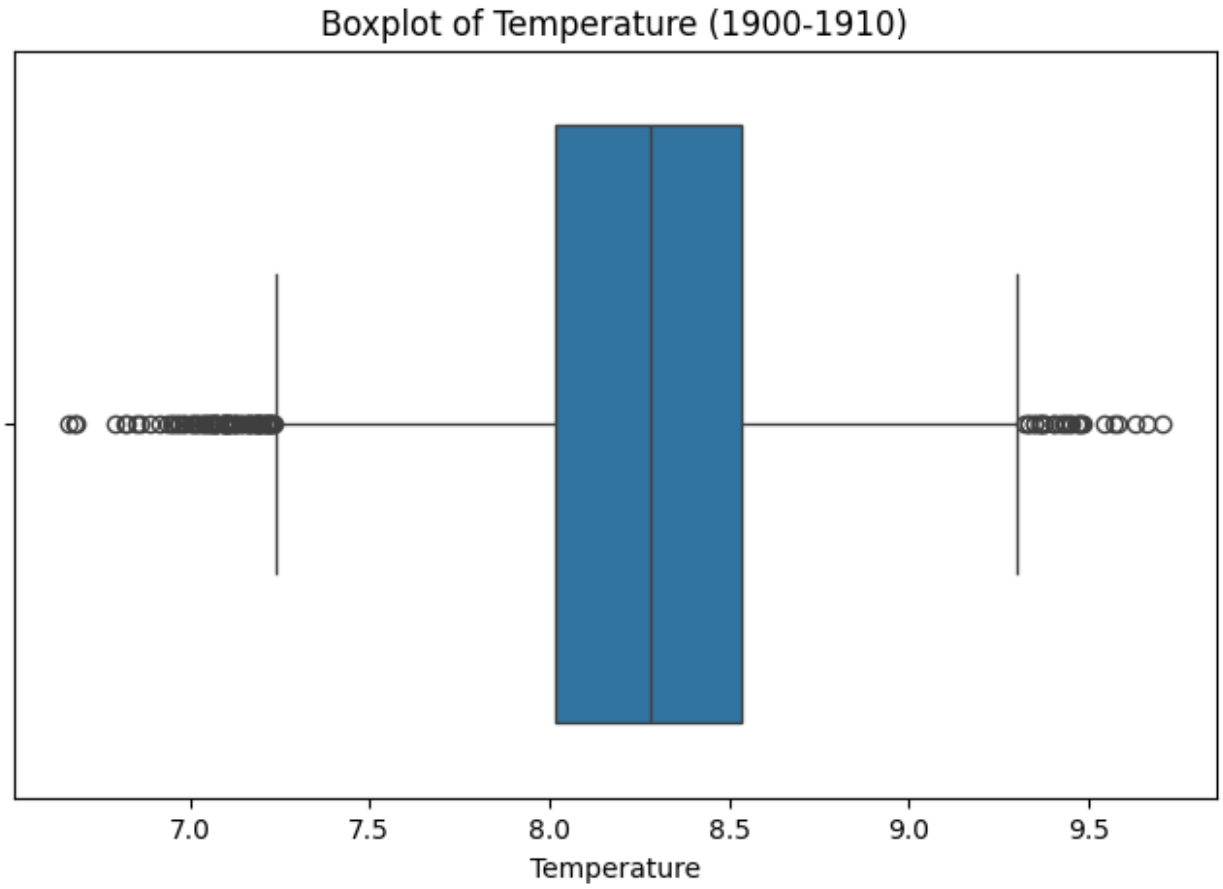

Boxplot of Temperature (1890-1900)
6.0
6.5
7.0
7.5
8.0
8.5
9.0
9.5
Temperature

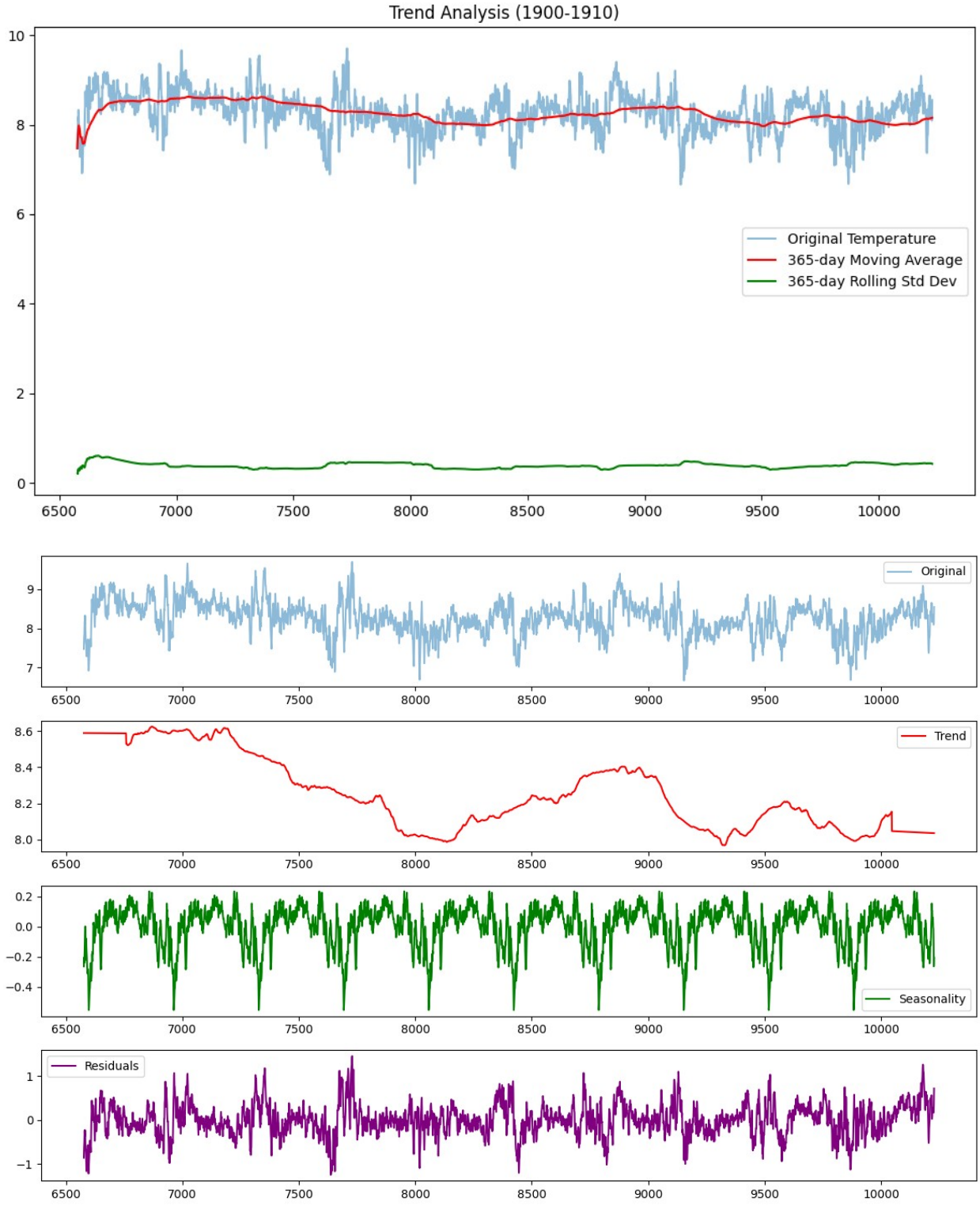

Trend Analysis (1890-1900)
10
8
6
4
2
0
3000
3500
4000
4500
5000
5500
6000
6500
Original Temperature
365-day Moving Average
365-day Rolling Std Dev

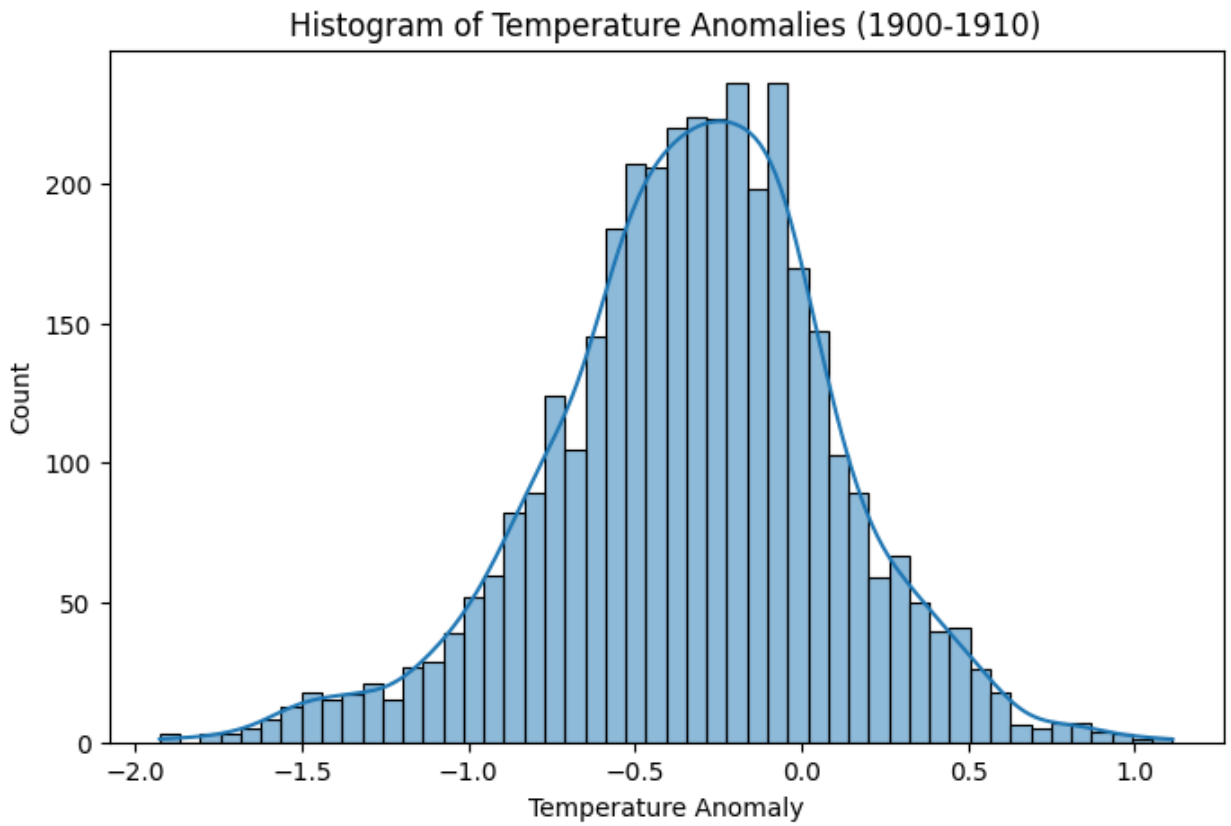

Original
Trend
Seasonality
Residuals
3000
3500
4000
4500
5000
5500
6000
6500


Histogram of Temperature Anomalies (1890-1900)
Count
Temperature Anomaly
0
50
100
150
200
250
300
350
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0

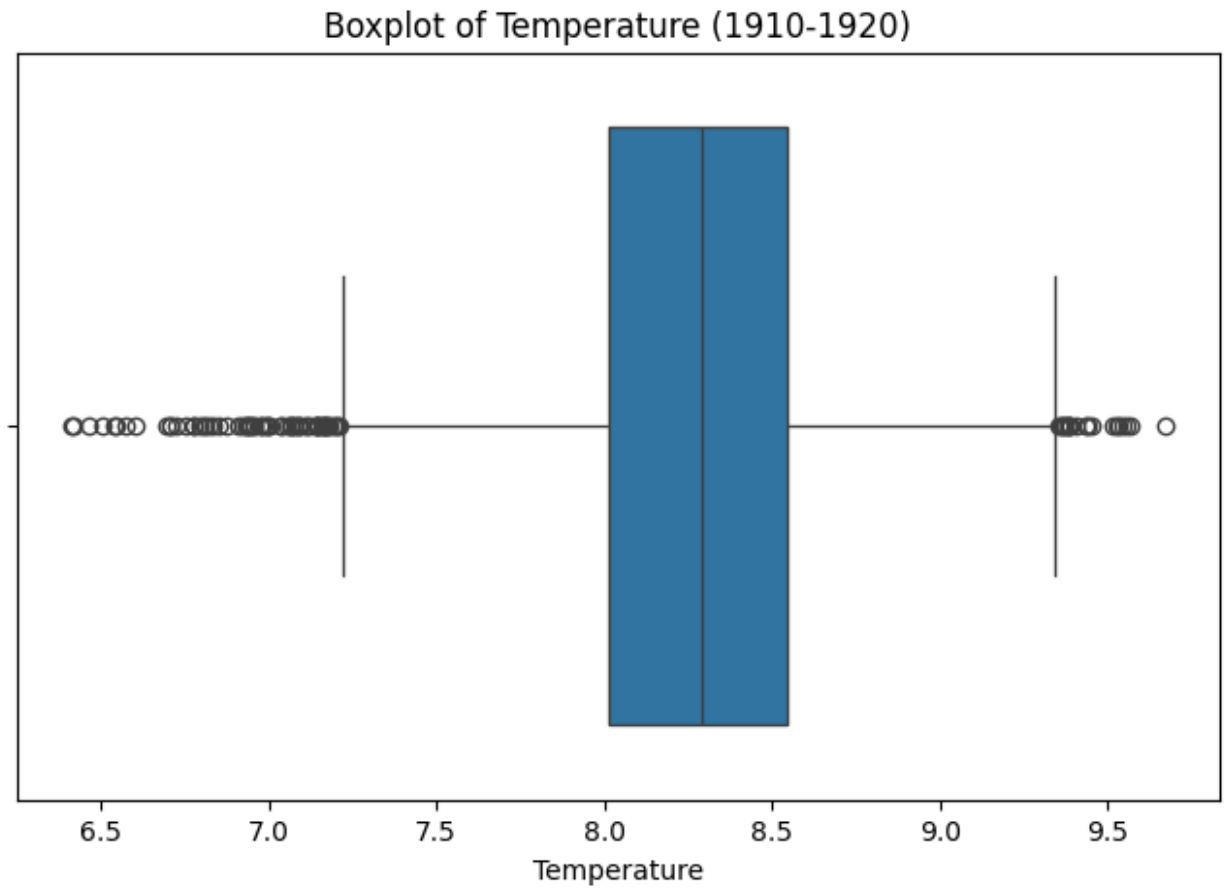
```
Analyzing period: 1900-1910
Number of Outliers: 105
```

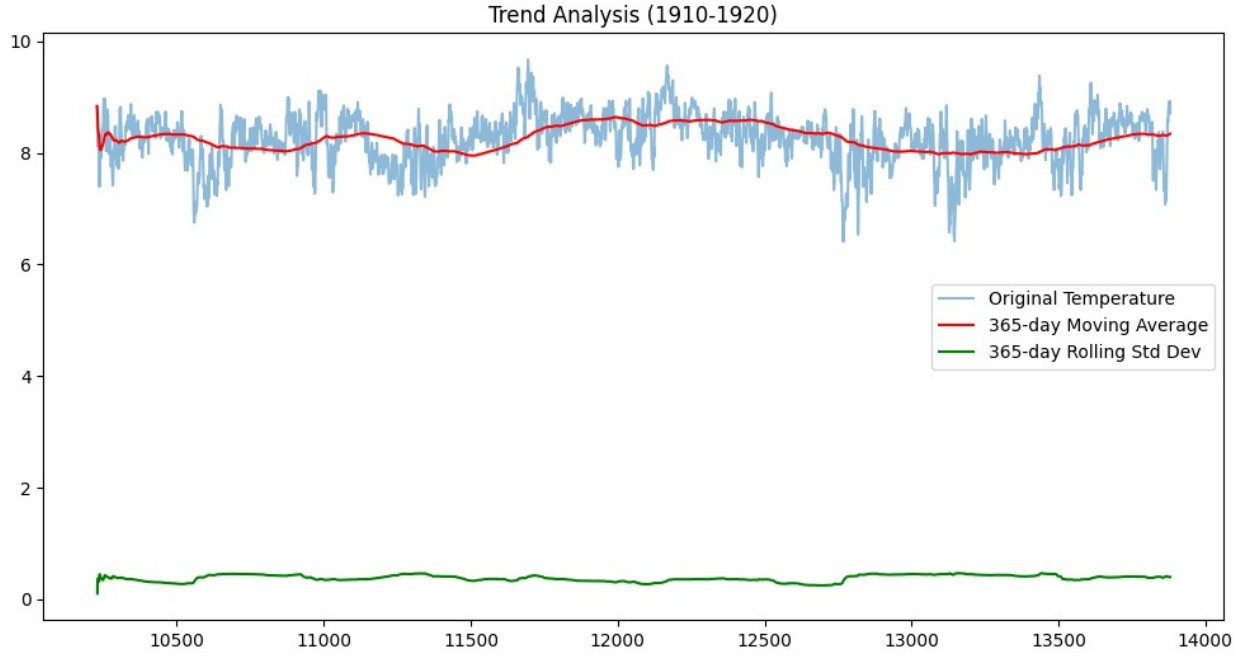


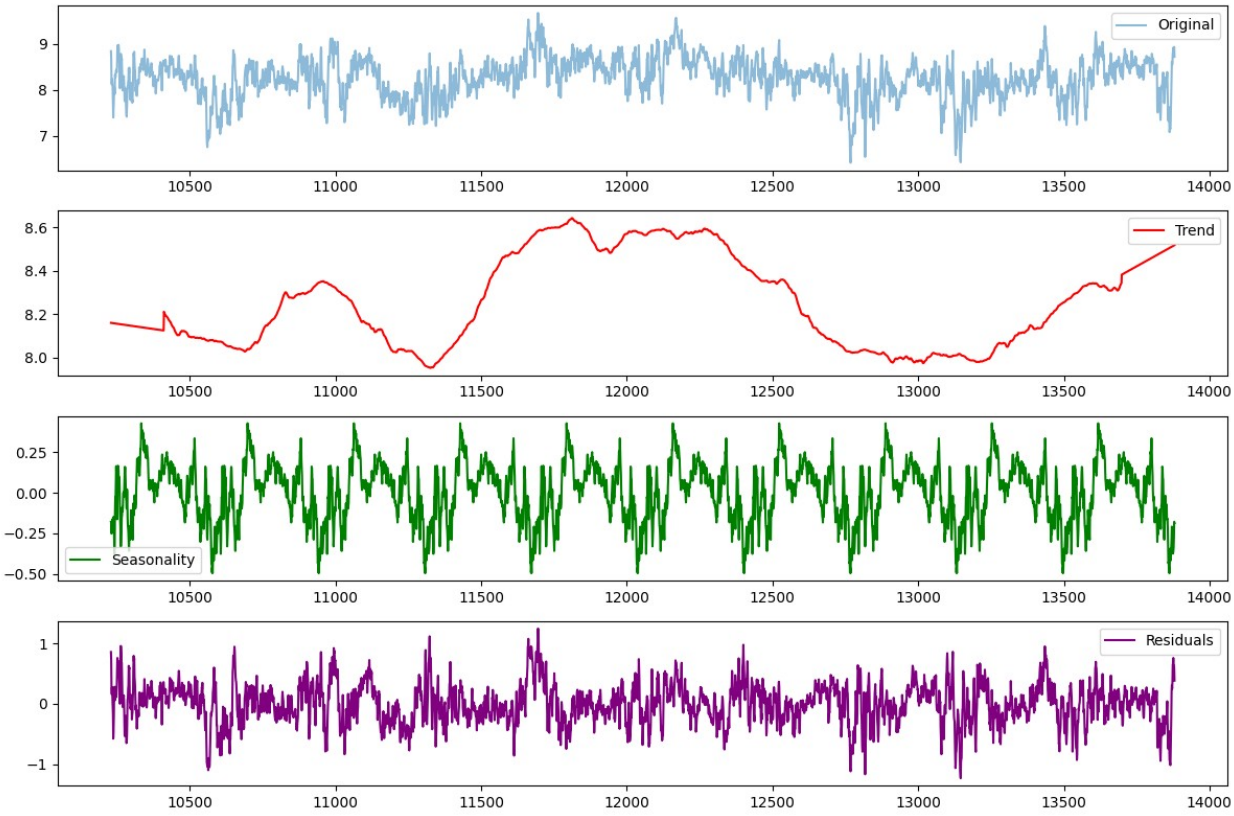

Trend Analysis (1900-1910)
Original Temperature
365-day Moving Average
365-day Rolling Std Dev
Original
Trend
Seasonality
Residuals

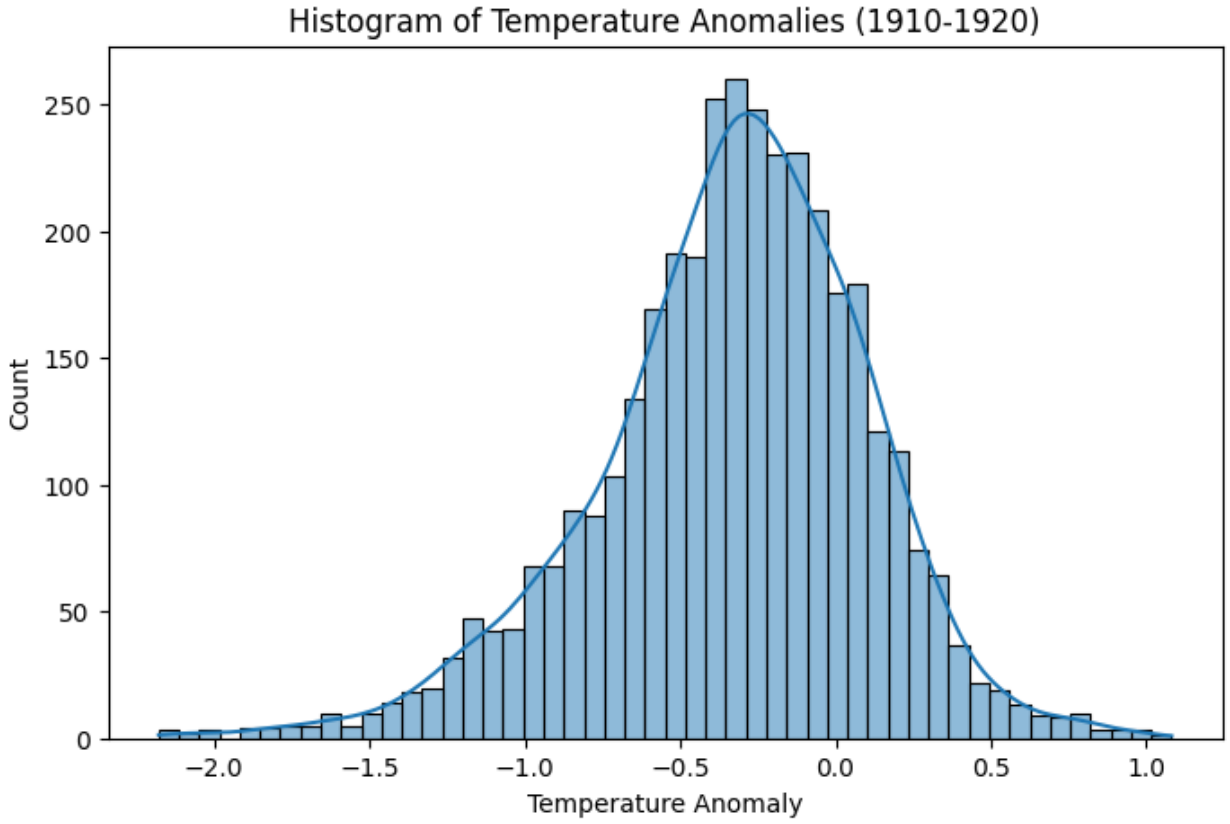



```
Analyzing period: 1910-1920
Number of Outliers: 88
```

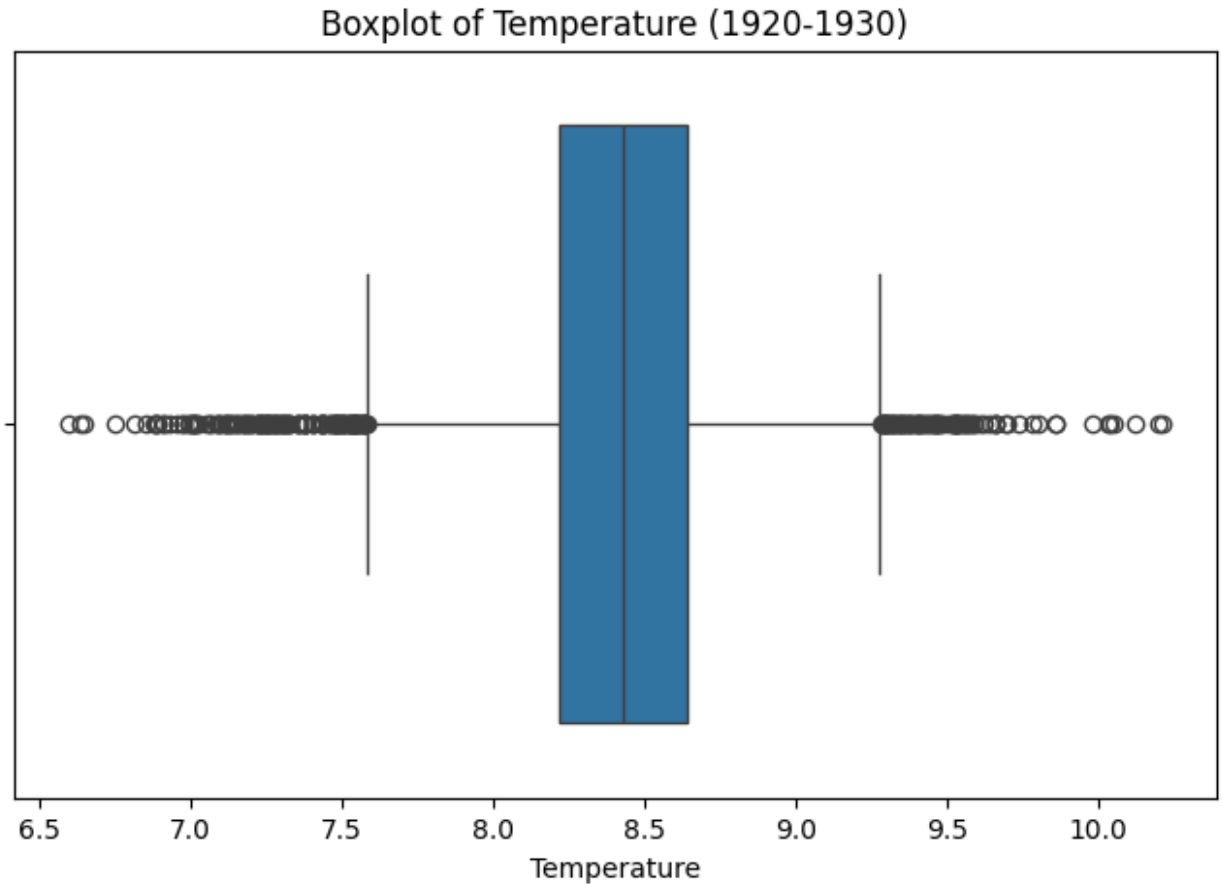

Boxplot of Temperature (1910-1920)
6.5
7.0
7.5
8.0
8.5
9.0
9.5
Temperature

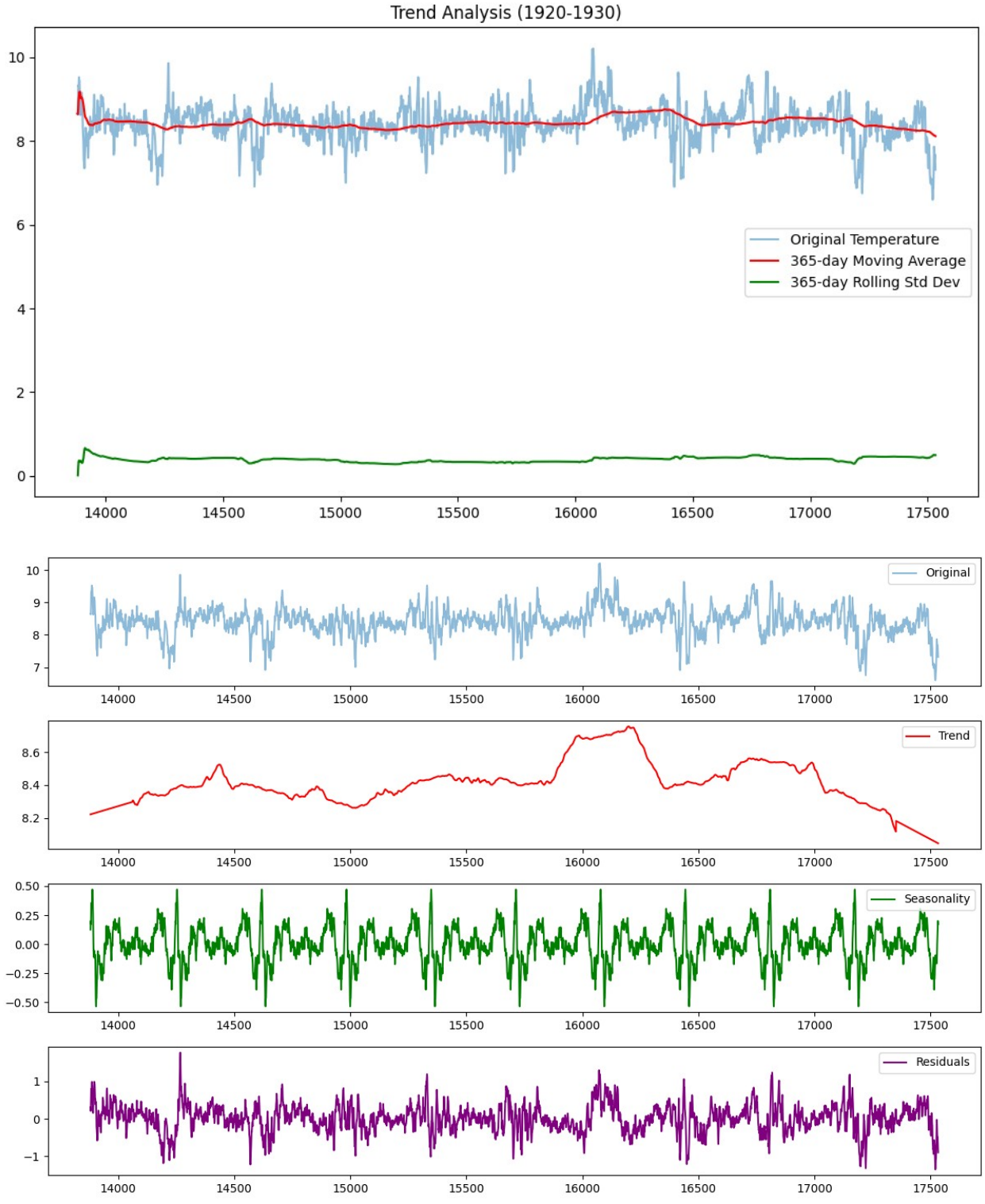

Trend Analysis (1910-1920)
10
8
6
4
2
0
10500
11000
11500
12000
12500
13000
13500
14000
Original Temperature
365-day Moving Average
365-day Rolling Std Dev

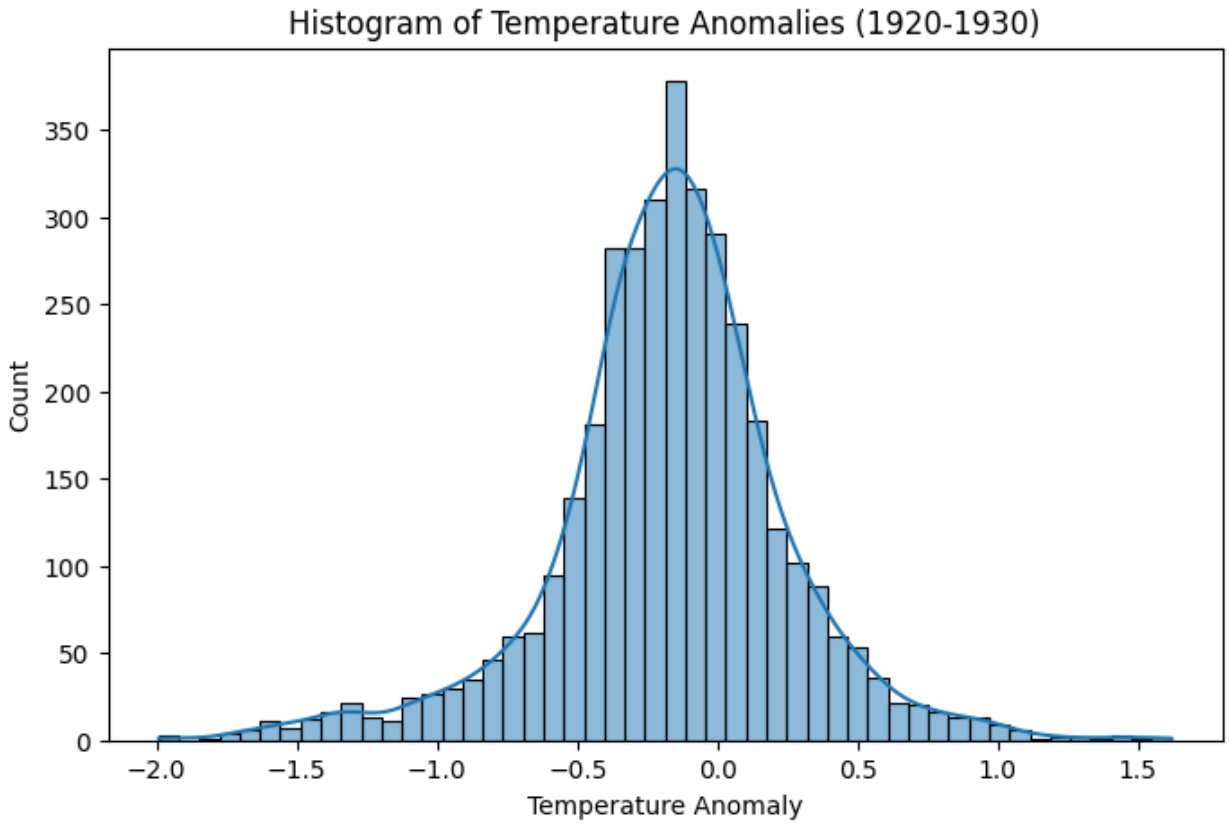

Original
Trend
Seasonality
Residuals
10500
11000
11500
12000
12500
13000
13500
14000


Histogram of Temperature Anomalies (1910-1920)
Count
Temperature Anomaly
0
50
100
150
200
250
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0

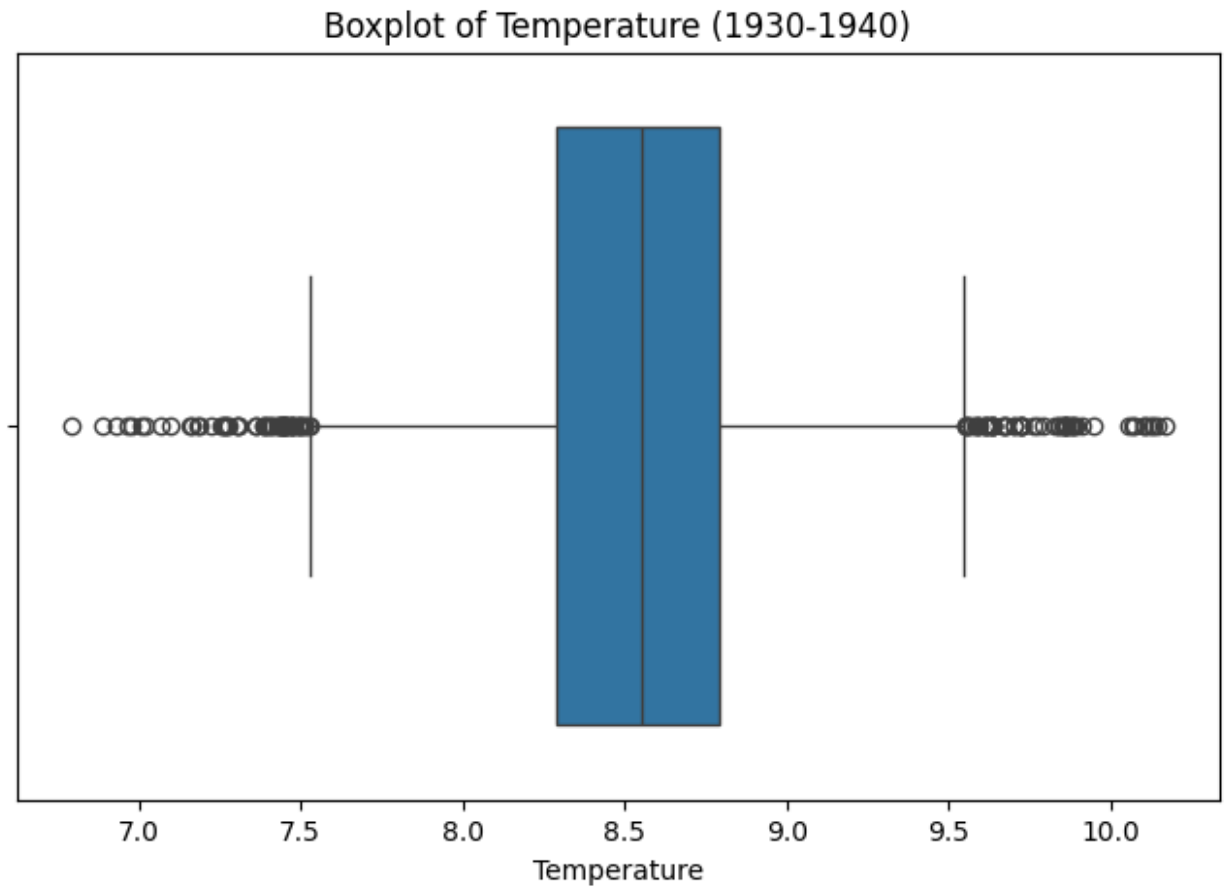
```
Analyzing period: 1920-1930
Number of Outliers: 236
```

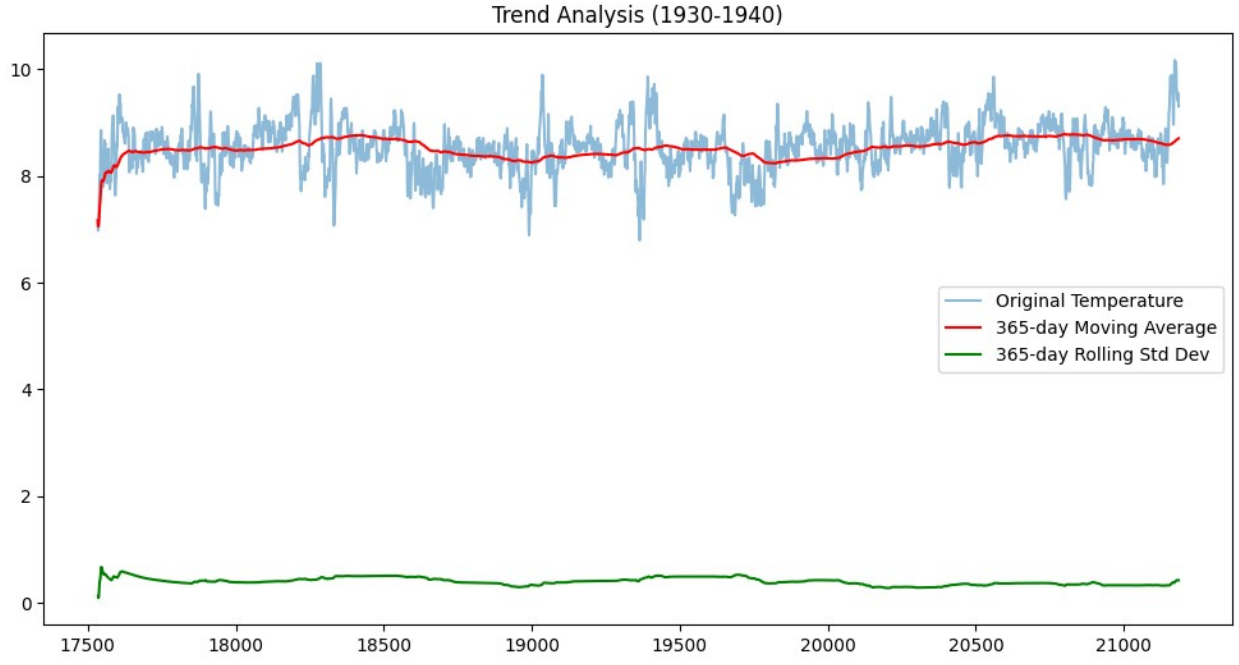


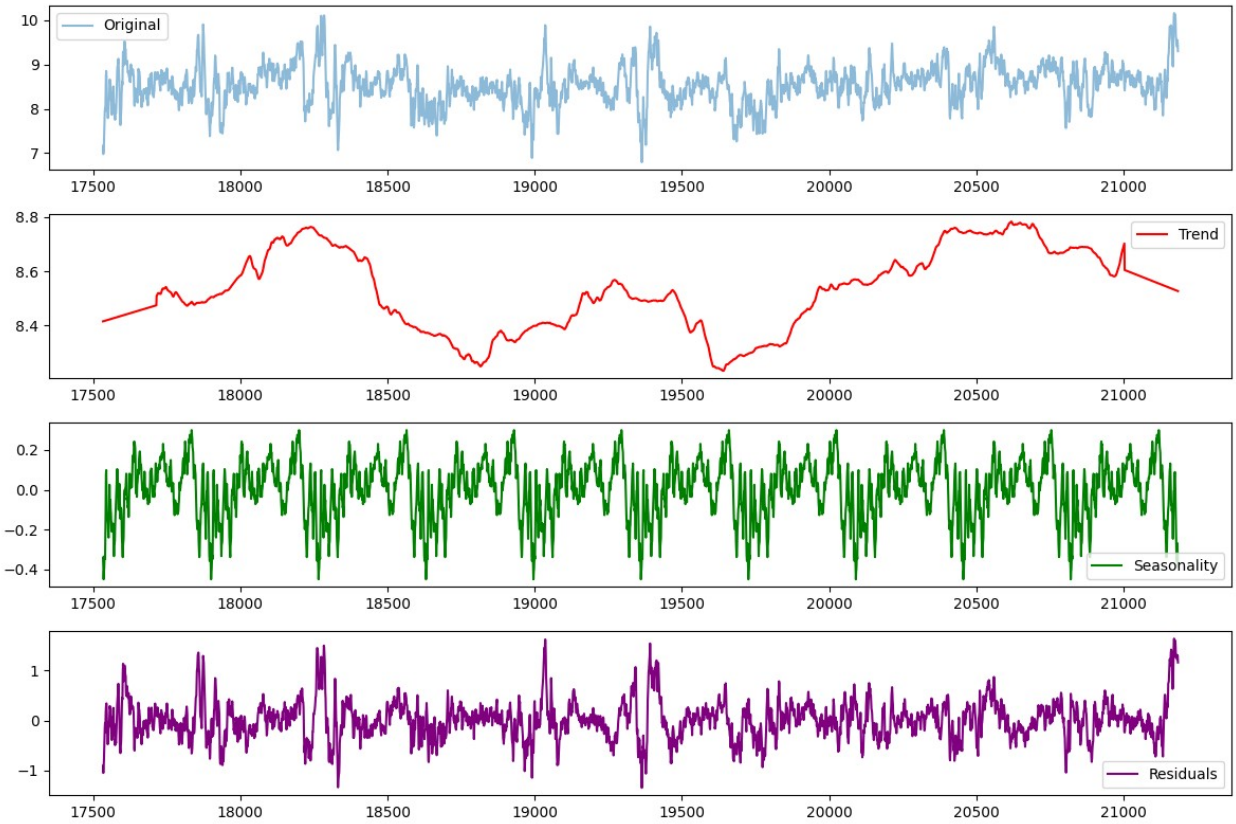

Trend Analysis (1920-1930)
Original Temperature
365-day Moving Average
365-day Rolling Std Dev
Original
Trend
Seasonality
Residuals

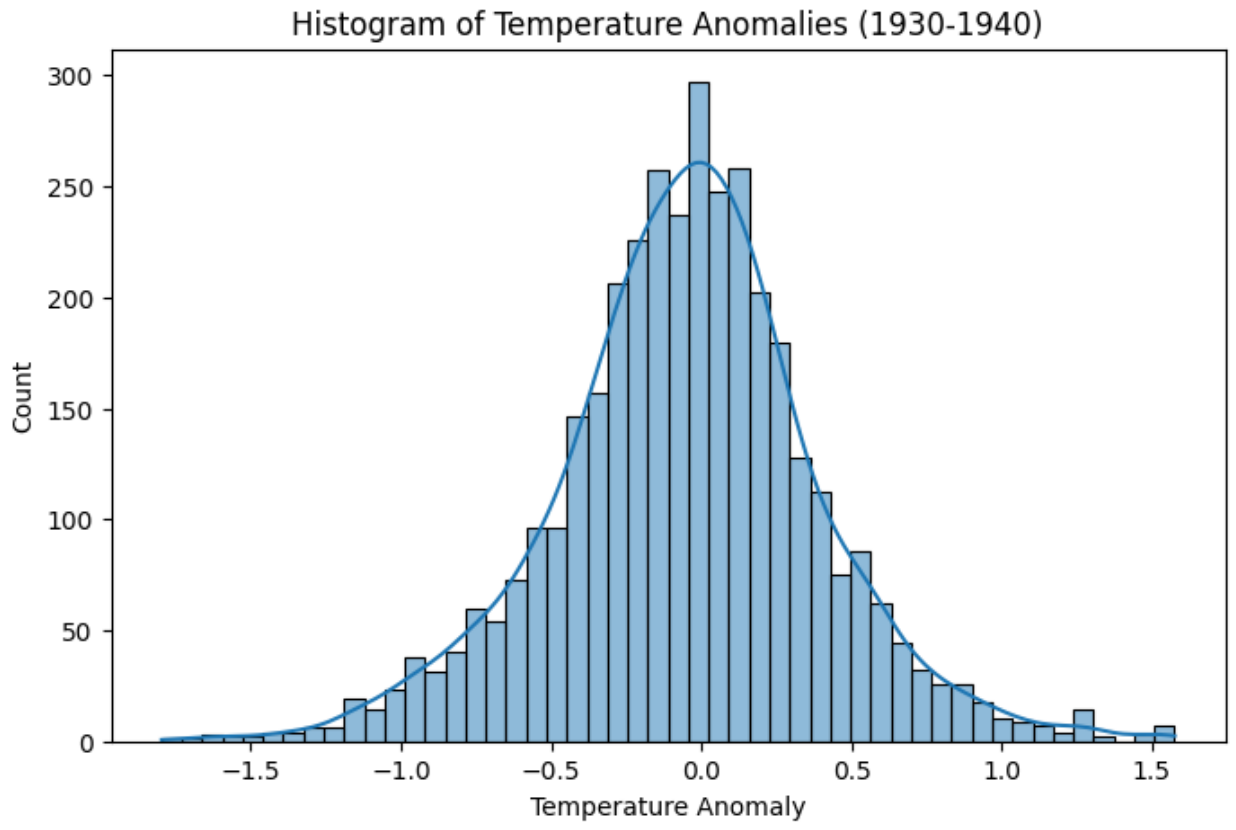



```
Analyzing period: 1930-1940
Number of Outliers: 121
```

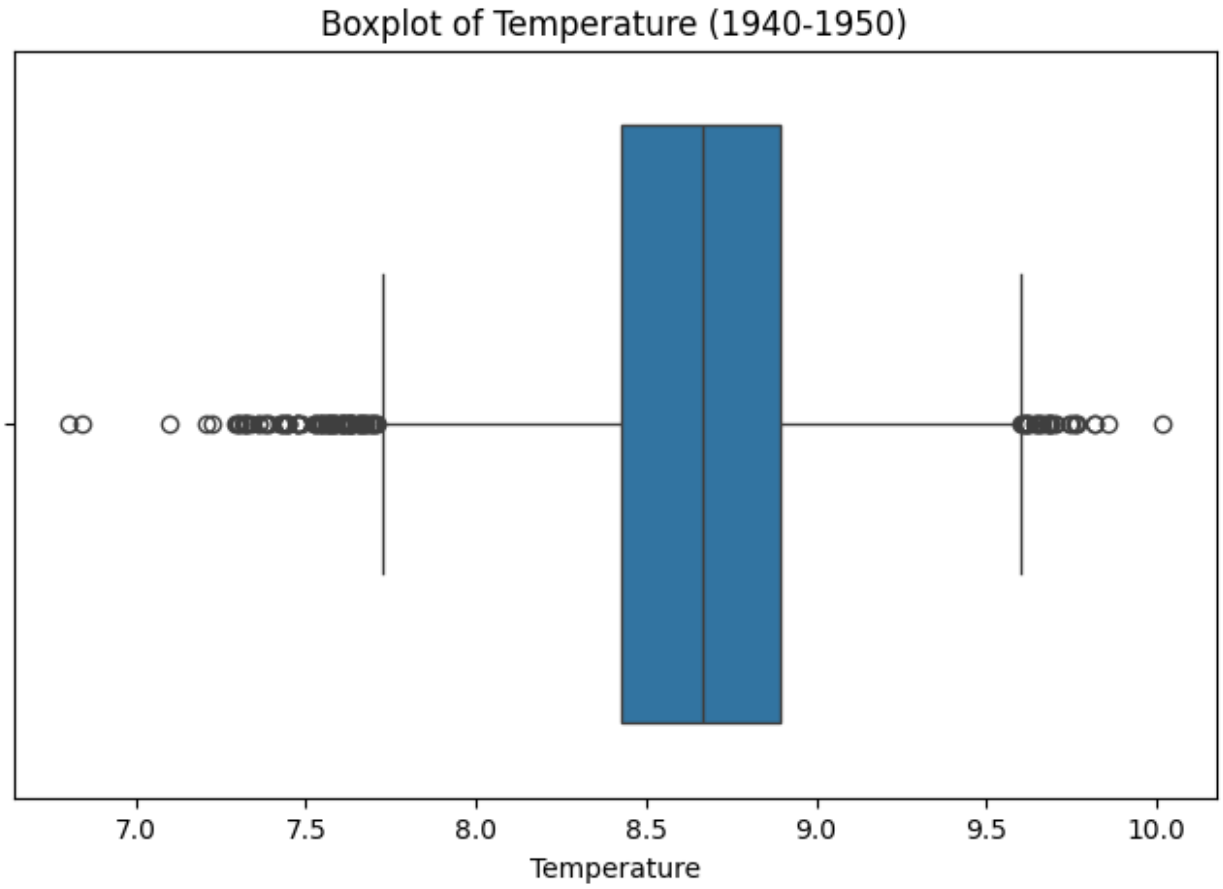

Boxplot of Temperature (1930-1940)
7.0
7.5
8.0
8.5
9.0
9.5
10.0
Temperature

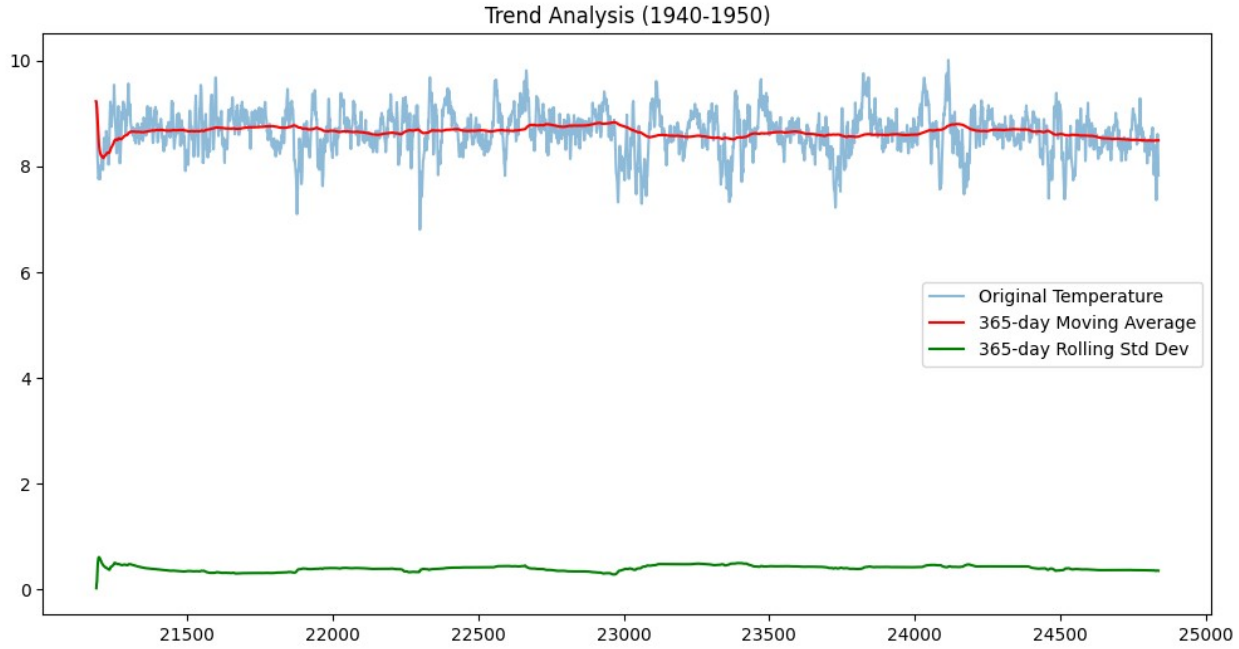

Trend Analysis (1930-1940)
10
8
6
4
2
0
17500
18000
18500
19000
19500
20000
20500
21000
Original Temperature
365-day Moving Average
365-day Rolling Std Dev

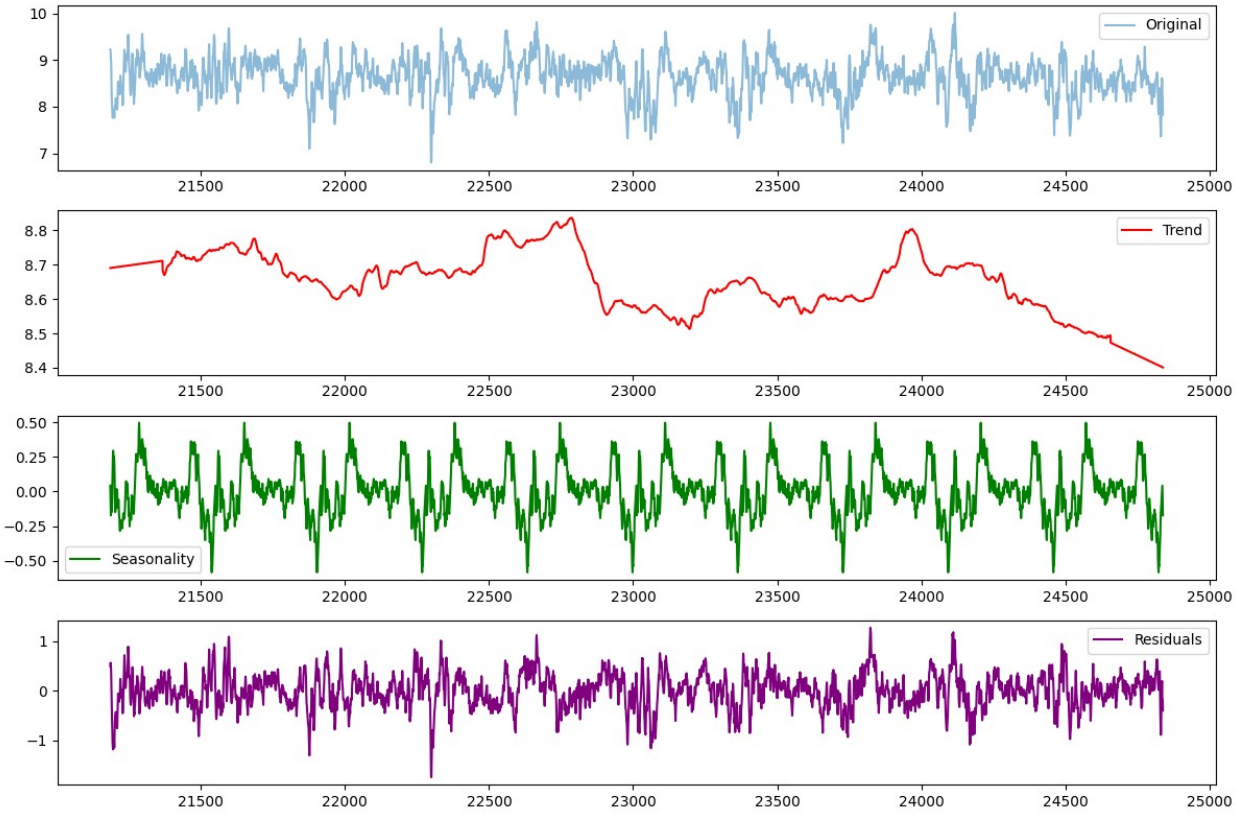

Original
Trend
Seasonality
Residuals
17500
18000
18500
19000
19500
20000
20500
21000

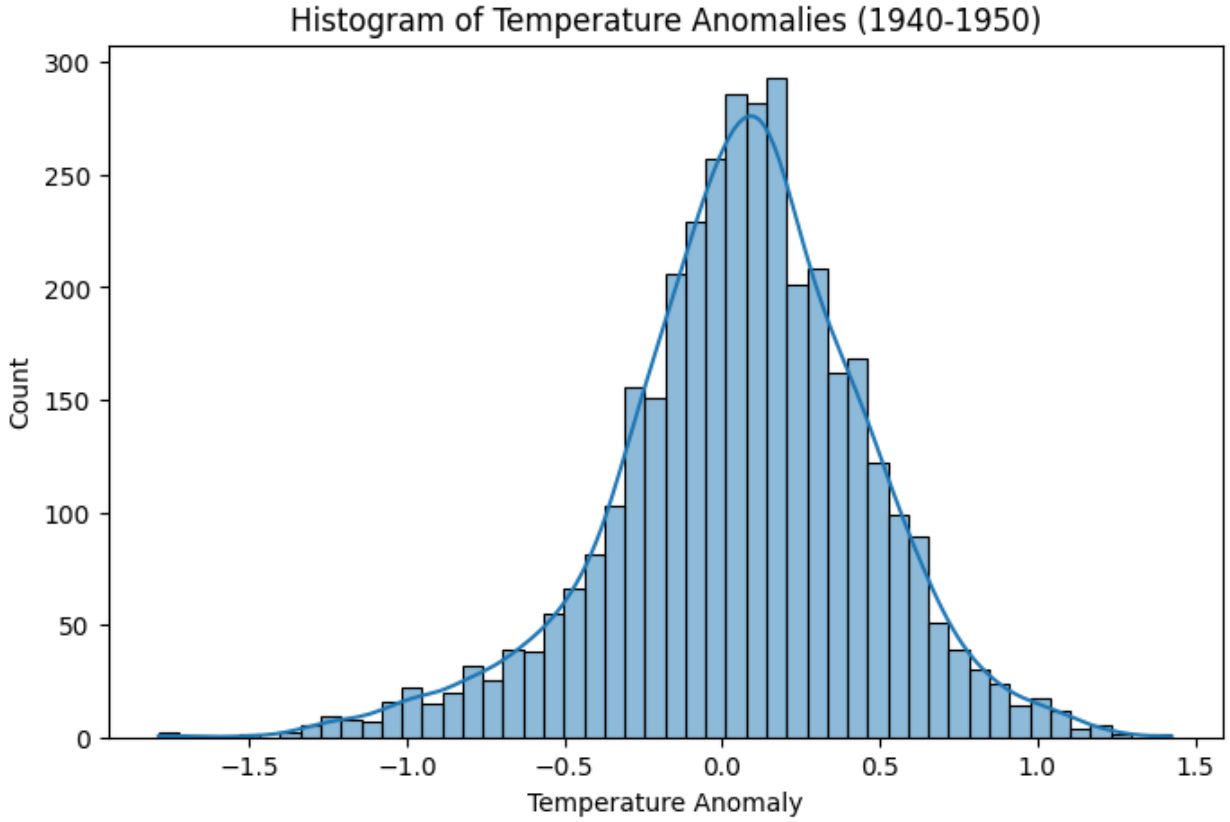

Histogram of Temperature Anomalies (1930-1940)
Count
Temperature Anomaly
0
50
100
150
200
250
300
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

```
Analyzing period: 1940-1950
Number of Outliers: 120
```

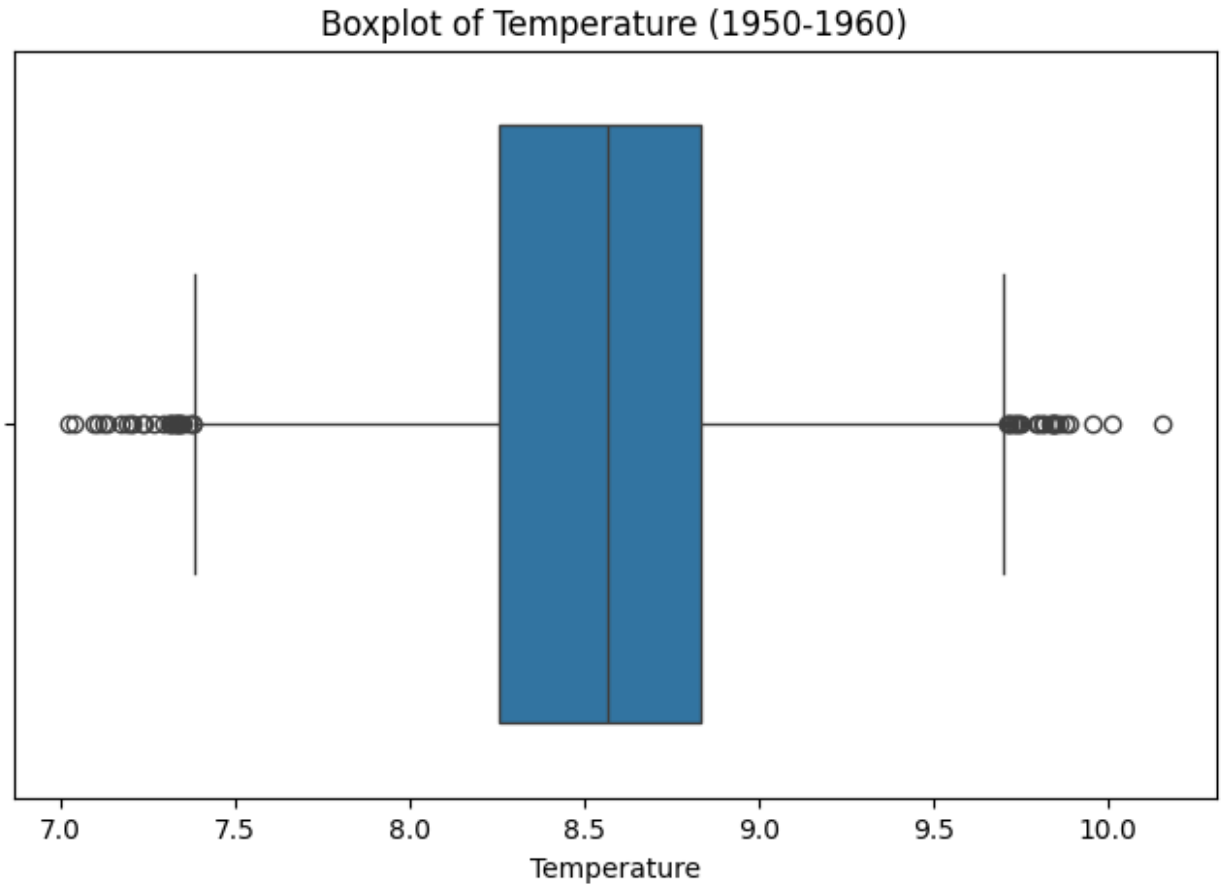


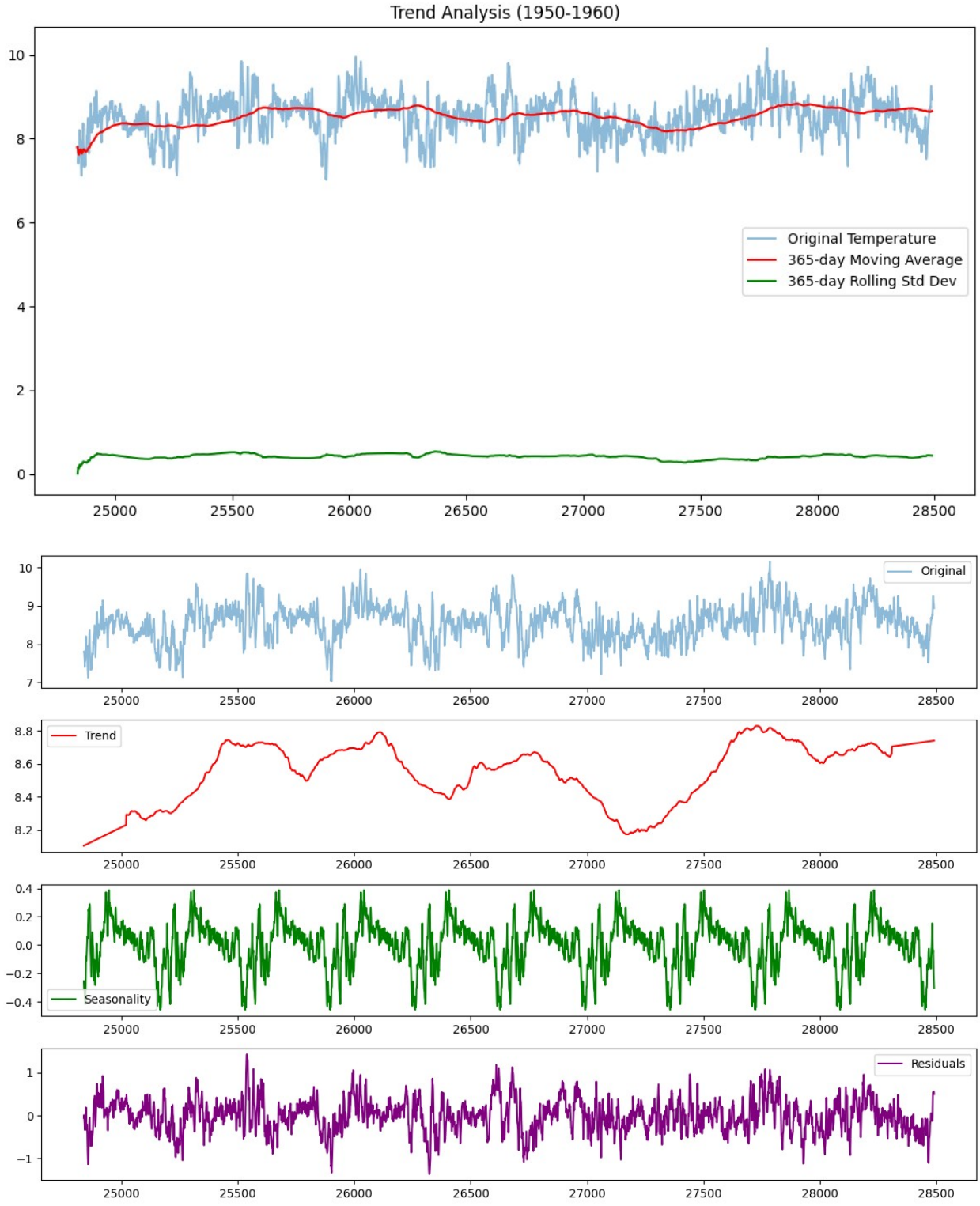


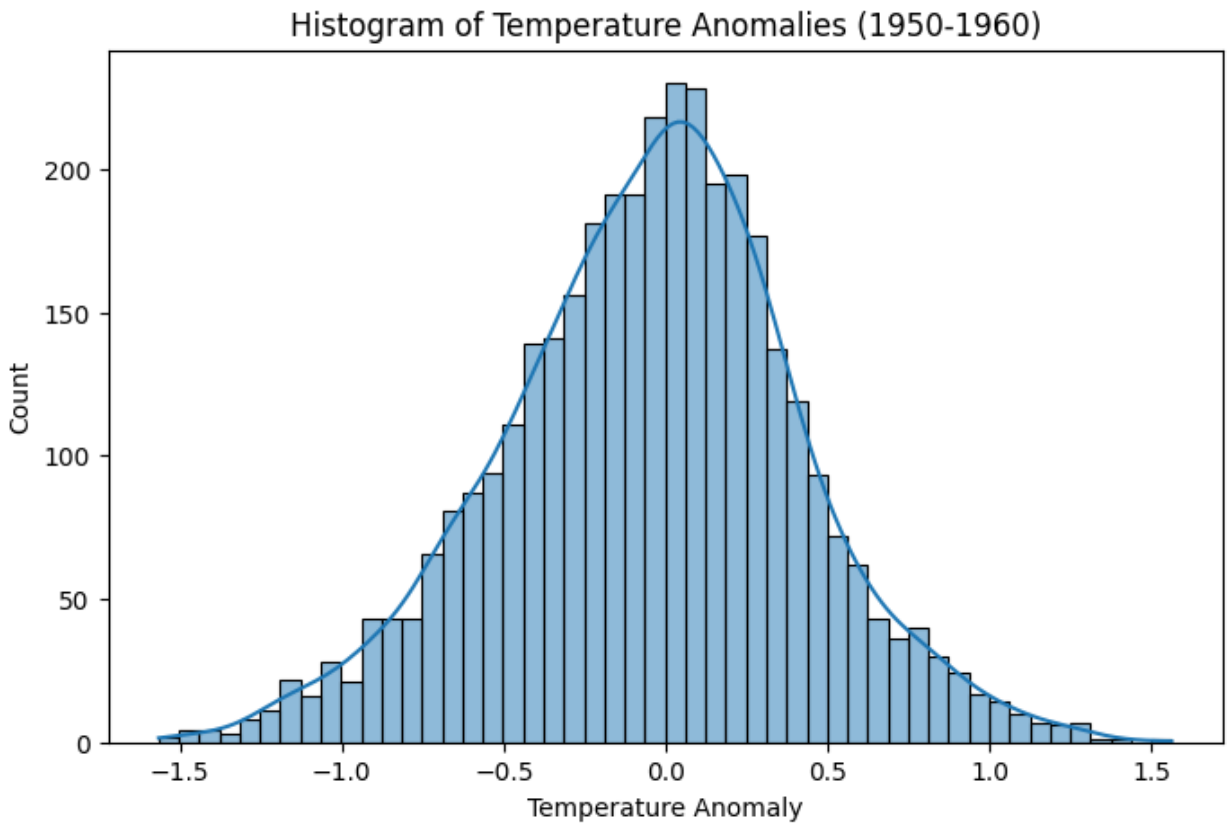

Original
Trend
Seasonality
Residuals
21500
22000
22500
23000
23500
24000
24500
25000


Histogram of Temperature Anomalies (1940-1950)
Count
Temperature Anomaly
0
50
100
150
200
250
300
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

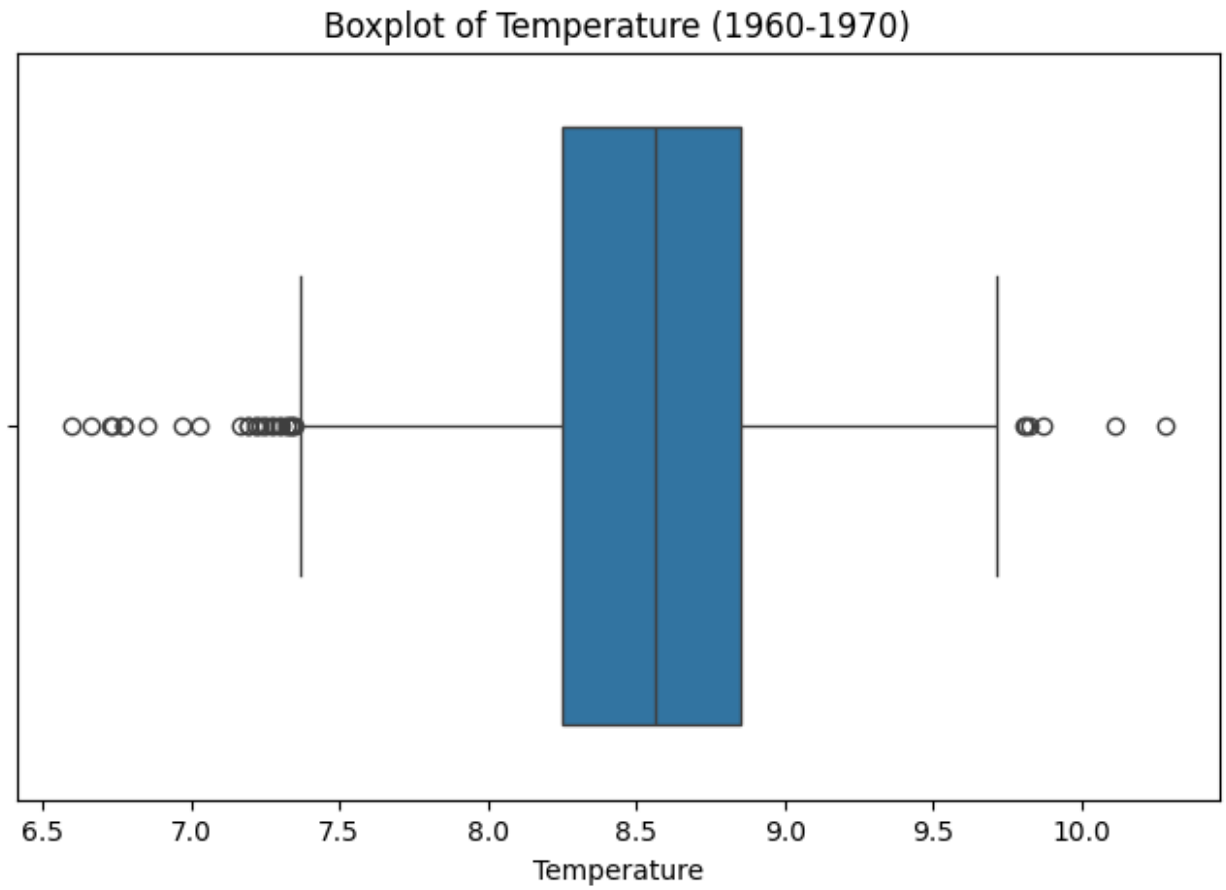
```
Analyzing period: 1950-1960
Number of Outliers: 52
```

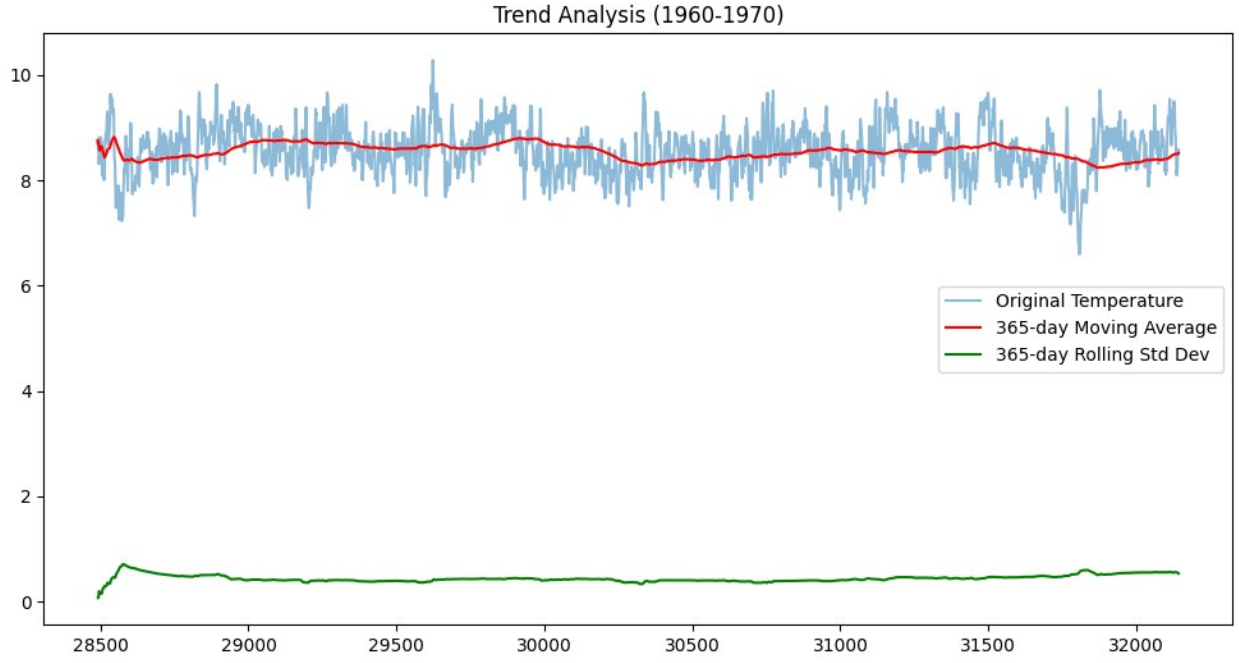


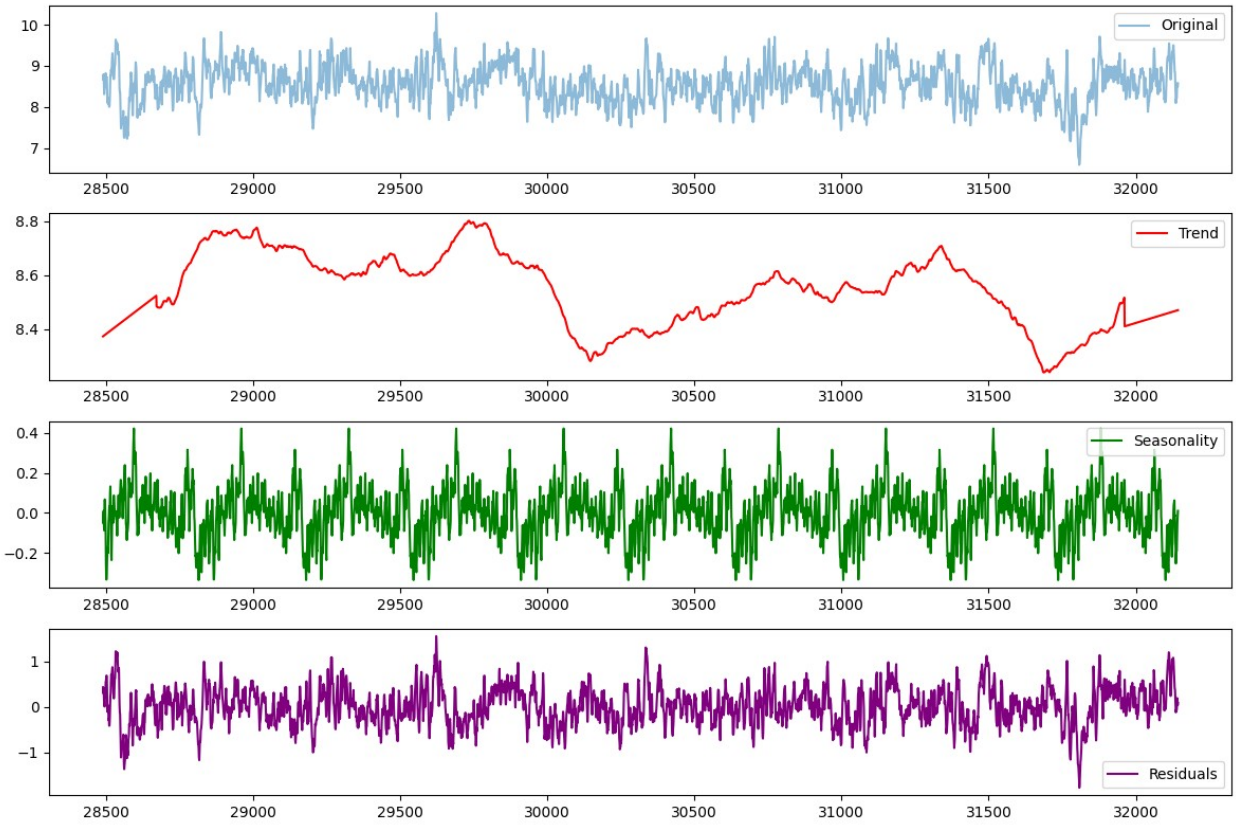

Trend Analysis (1950-1960)
Original Temperature
365-day Moving Average
365-day Rolling Std Dev
Original
Trend
Seasonality
Residuals

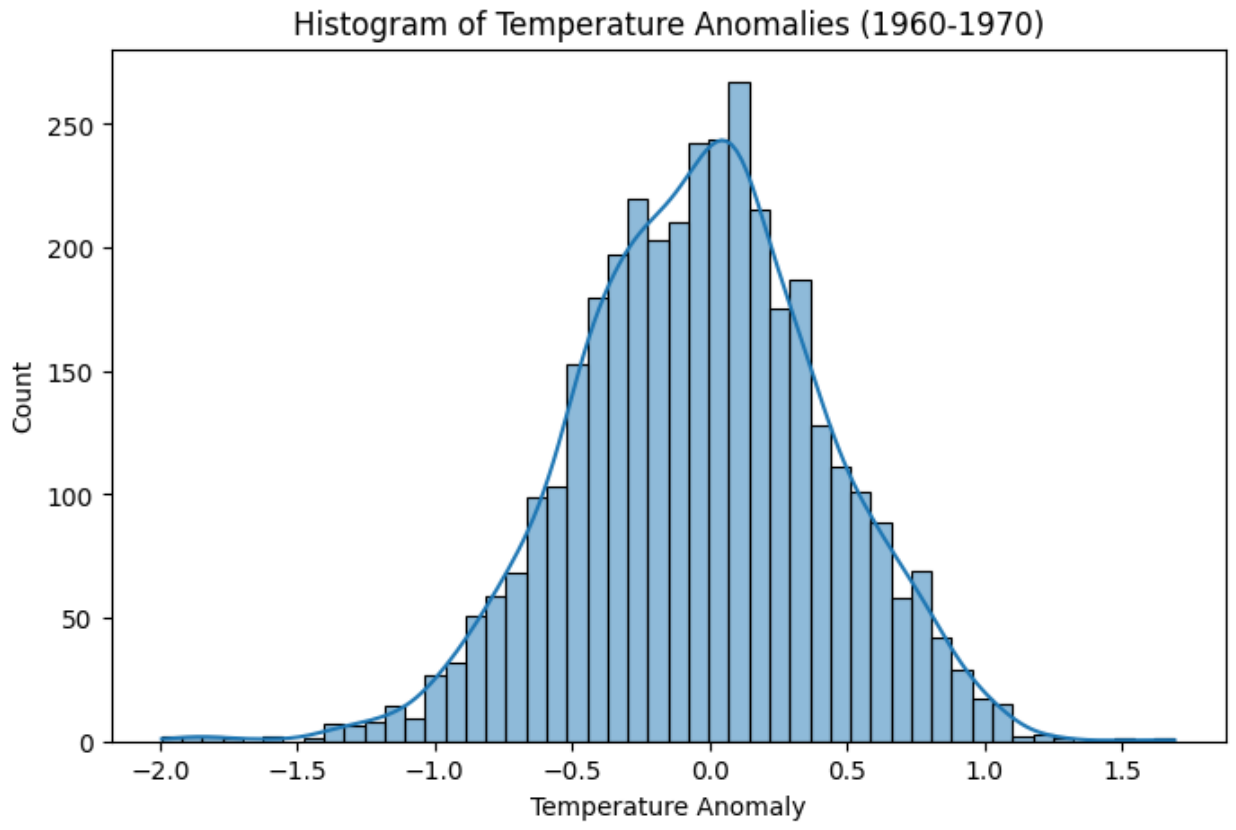



```
Analyzing period: 1960-1970
Number of Outliers: 33
```

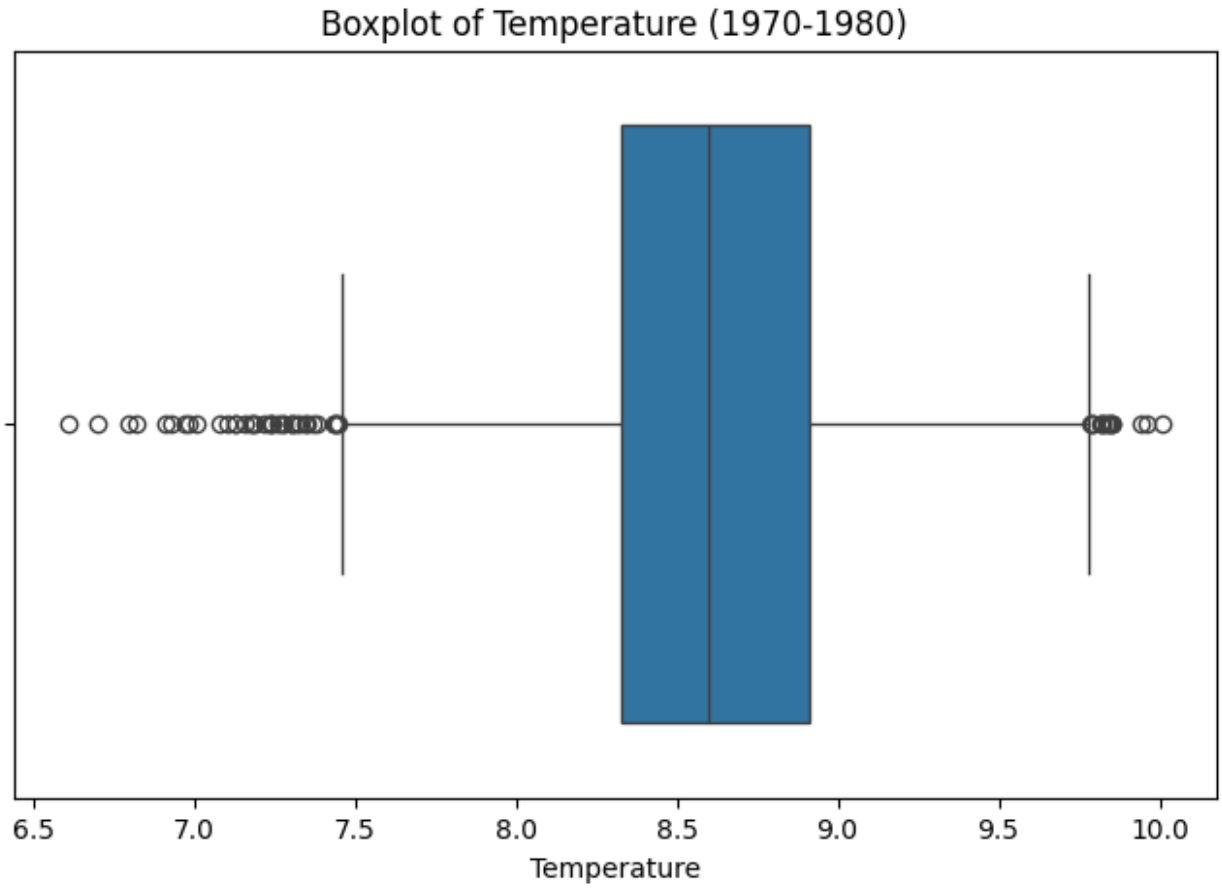

Boxplot of Temperature (1960-1970)
6.5
7.0
7.5
8.0
8.5
9.0
9.5
10.0
Temperature

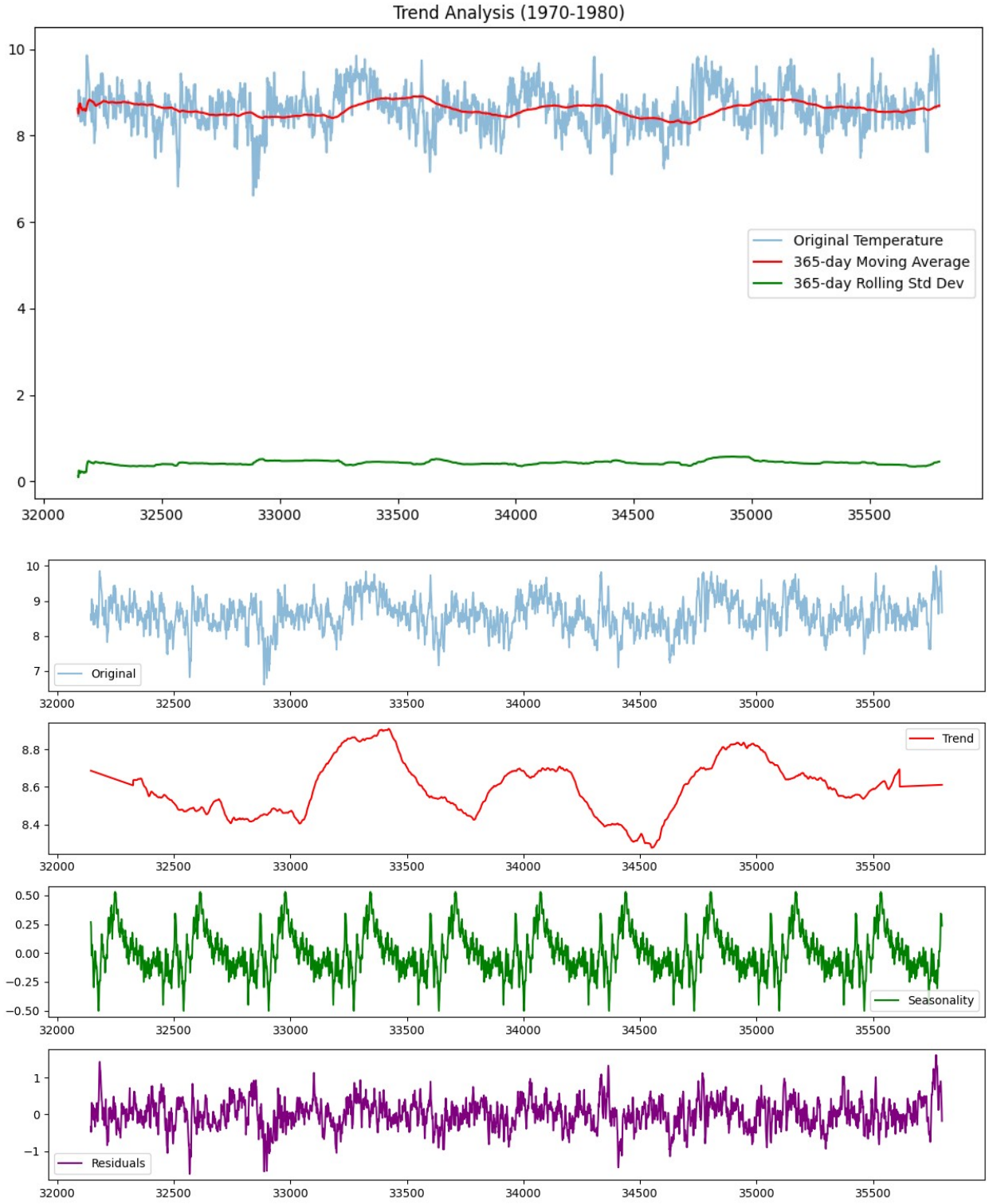

Trend Analysis (1960-1970)
10
8
6
4
2
0
28500
29000
29500
30000
30500
31000
31500
32000
Original Temperature
365-day Moving Average
365-day Rolling Std Dev

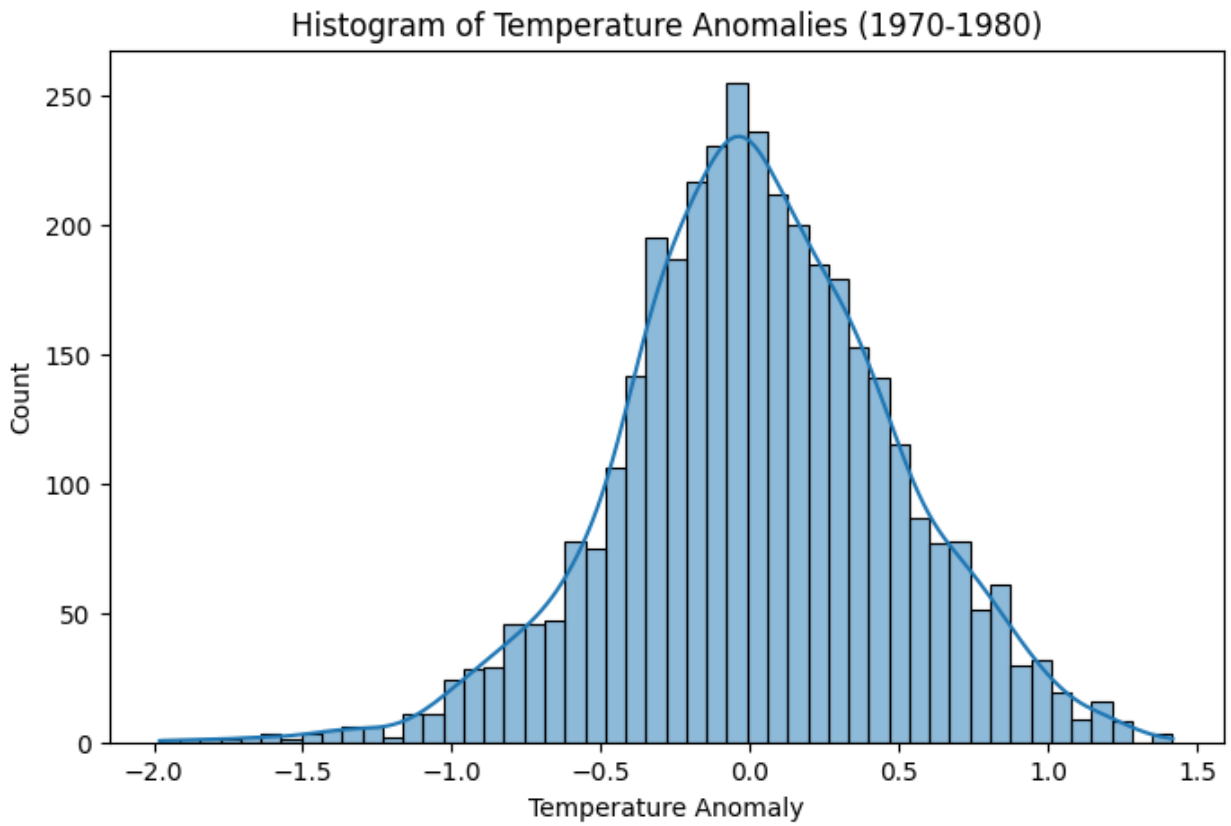

Original
Trend
Seasonality
Residuals
28500
29000
29500
30000
30500
31000
31500
32000


Histogram of Temperature Anomalies (1960-1970)
Count
Temperature Anomaly
0
50
100
150
200
250
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

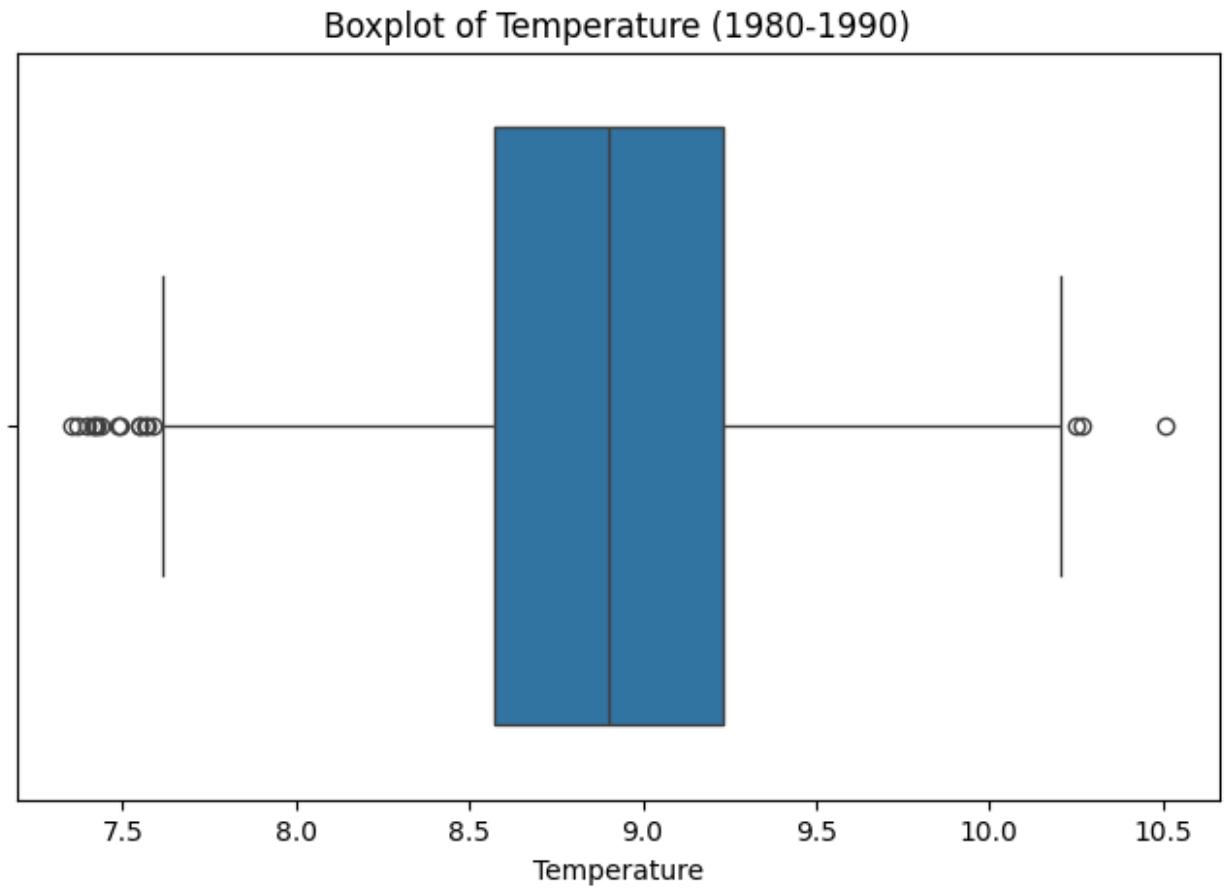
```
Analyzing period: 1970-1980
Number of Outliers: 50
```

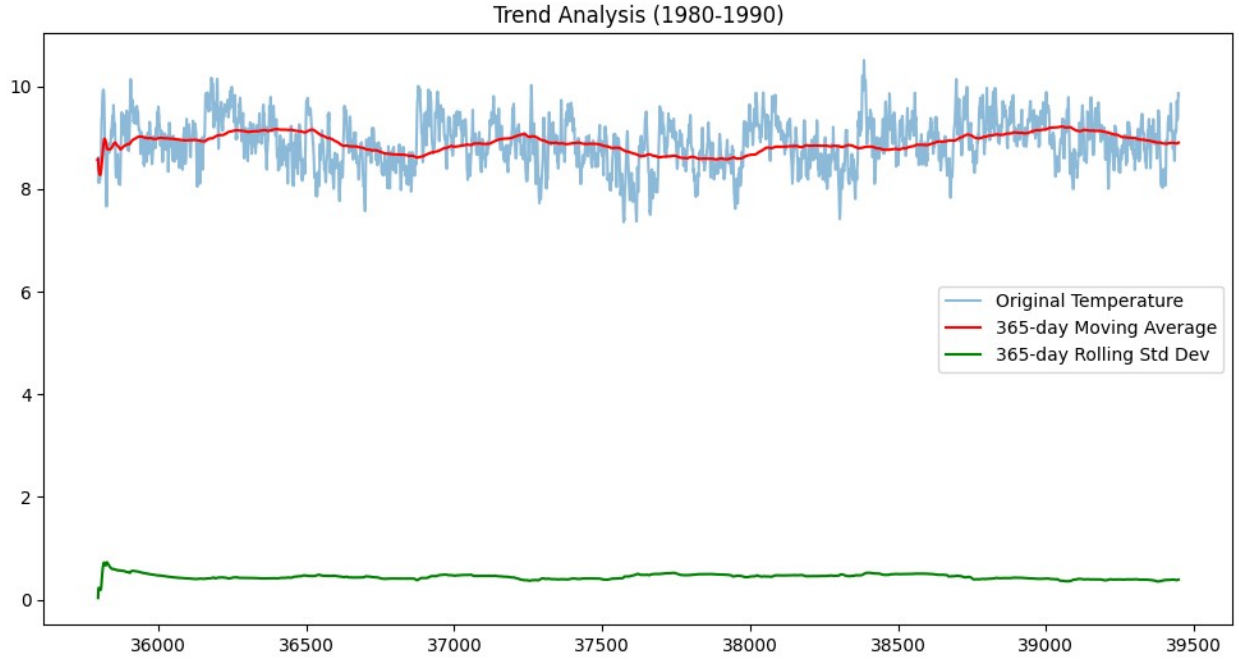


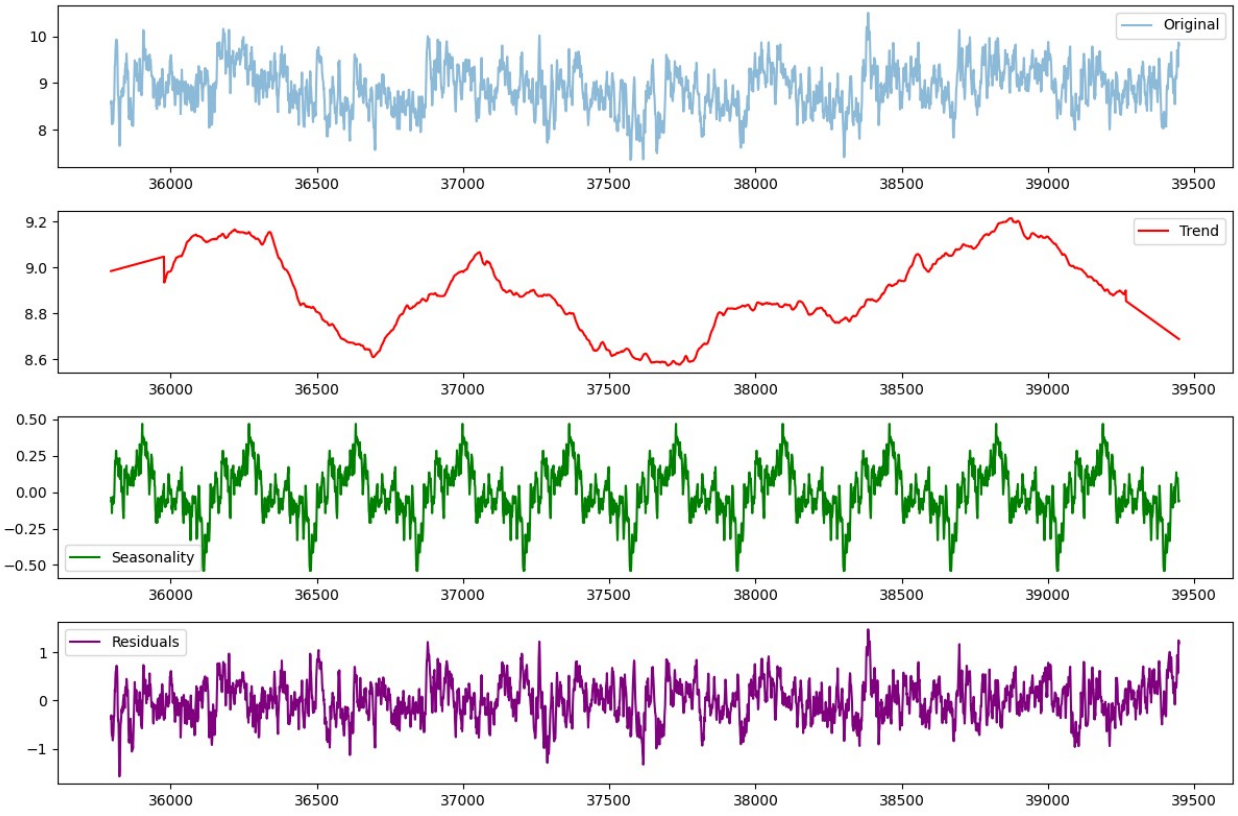

Trend Analysis (1970-1980)
Original Temperature
365-day Moving Average
365-day Rolling Std Dev
Original
Trend
Seasonality
Residuals

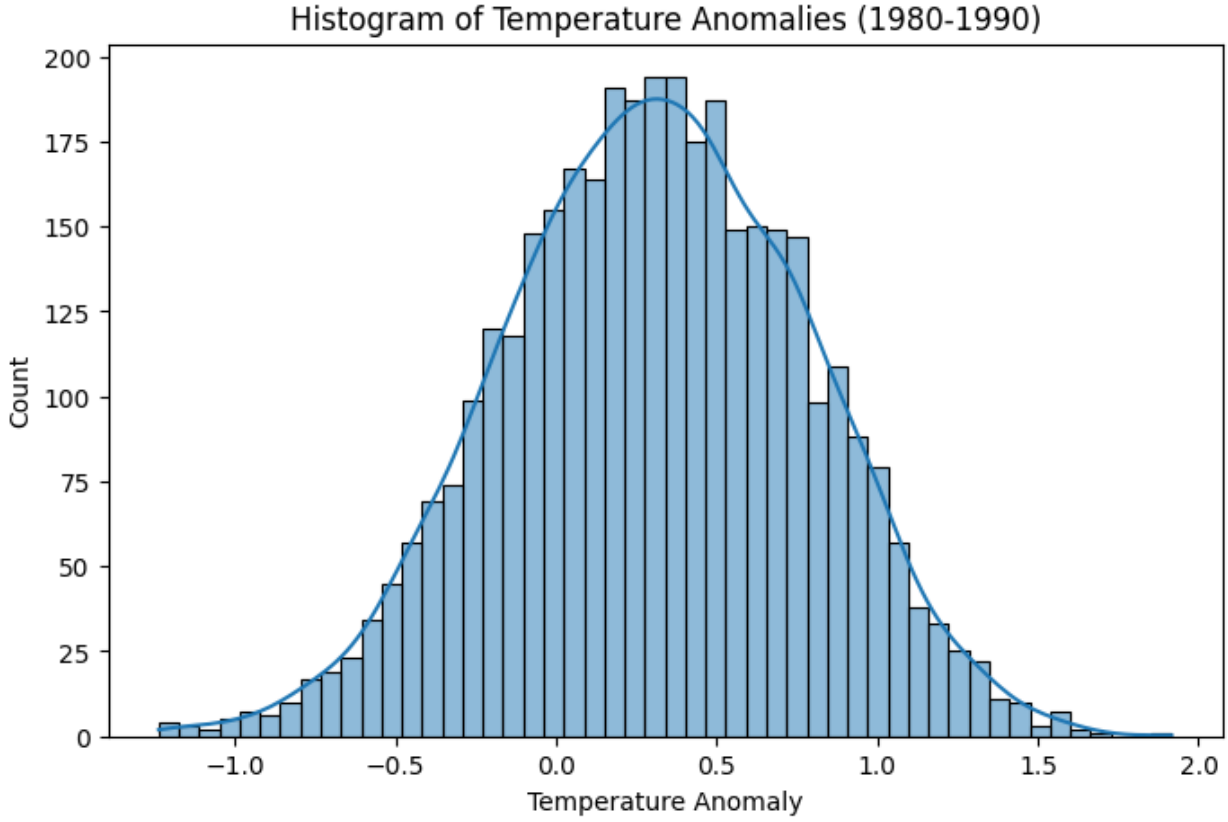



```
Analyzing period: 1980-1990
Number of Outliers: 17
```

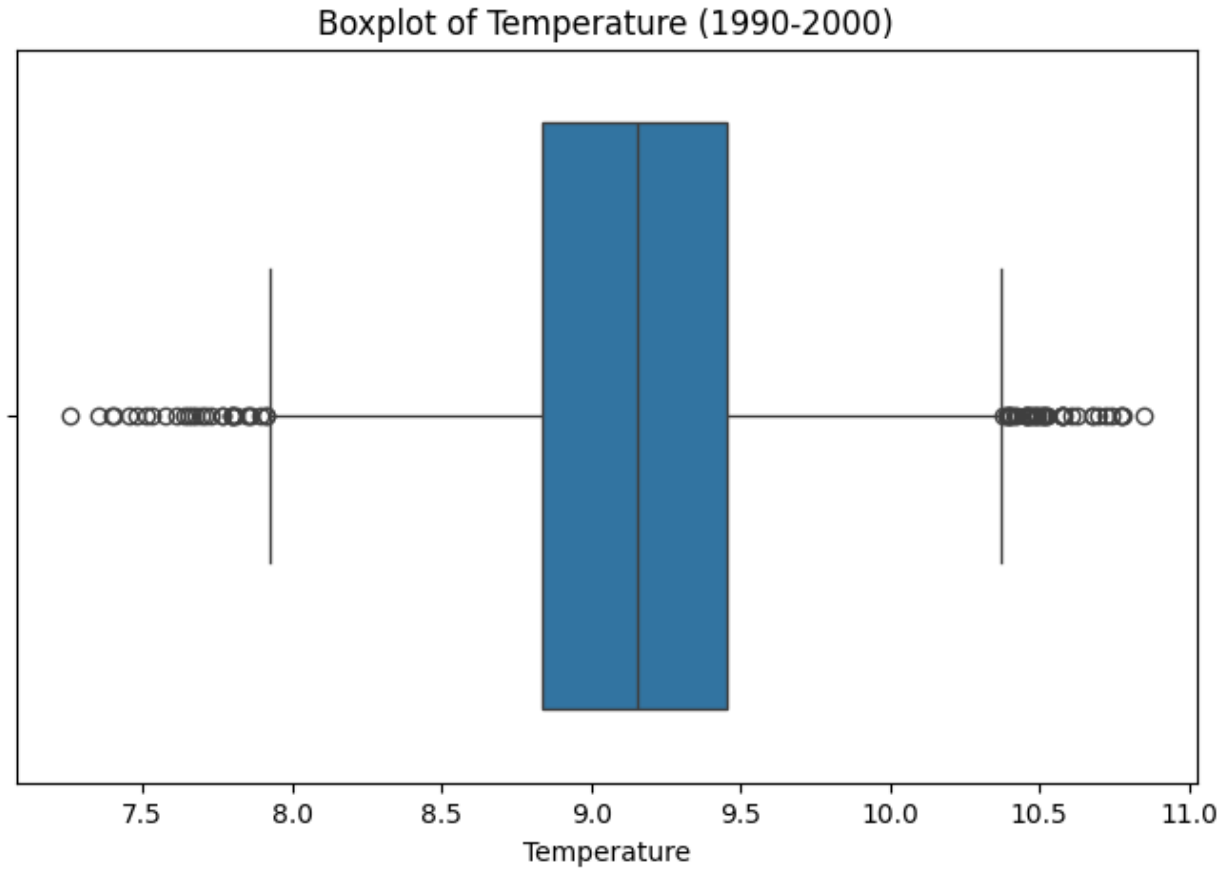

Boxplot of Temperature (1980-1990)
7.5
8.0
8.5
9.0
9.5
10.0
10.5
Temperature

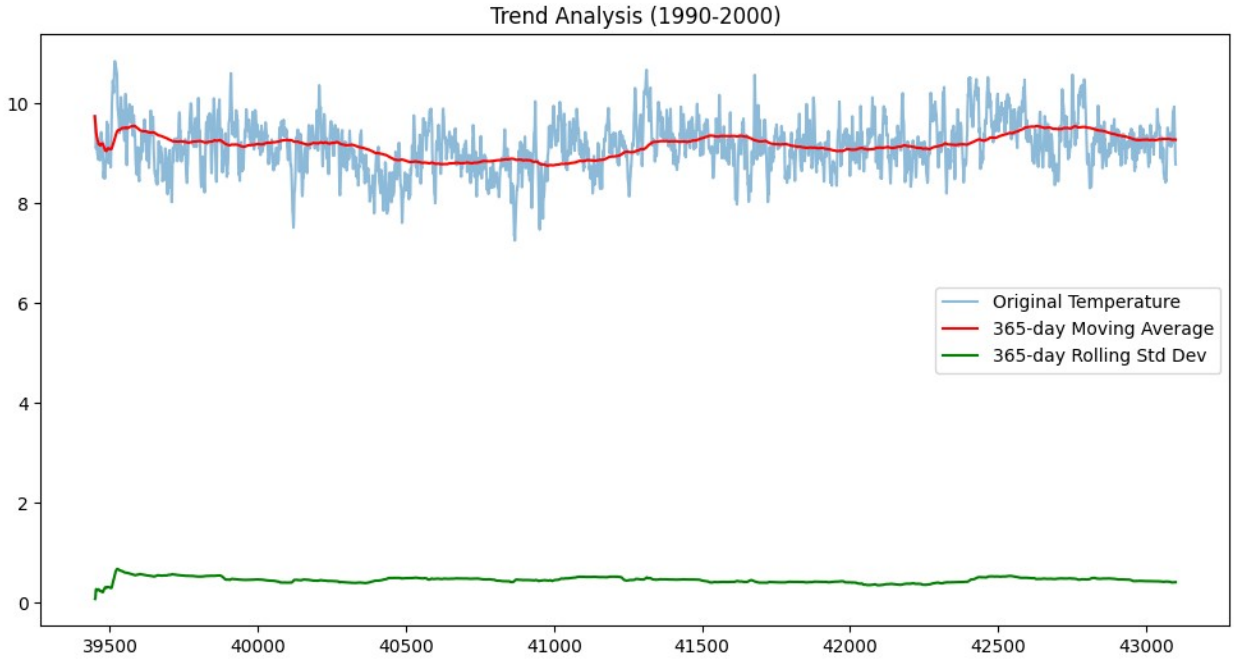

Trend Analysis (1980-1990)
10
8
6
4
2
0
36000
36500
37000
37500
38000
38500
39000
39500
Original Temperature
365-day Moving Average
365-day Rolling Std Dev

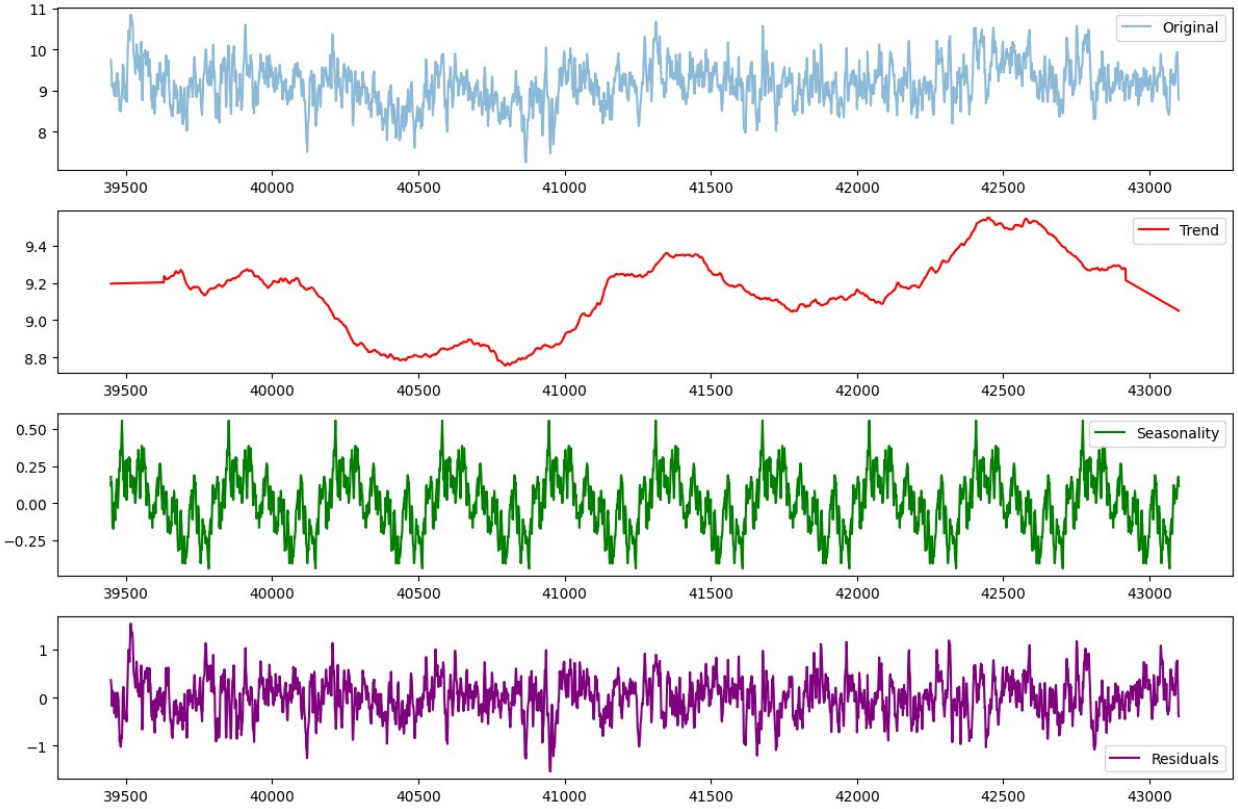

Original
Trend
Seasonality
Residuals
36000
36500
37000
37500
38000
38500
39000
39500

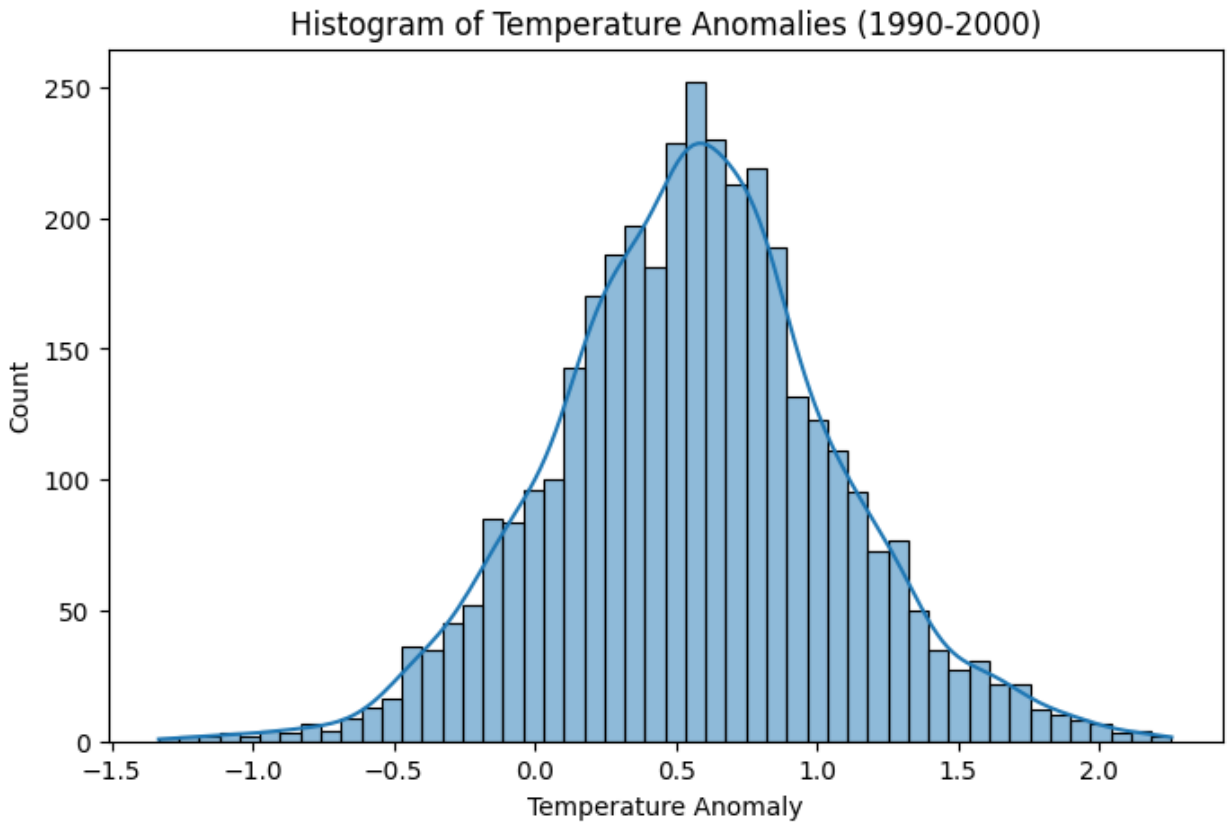

Histogram of Temperature Anomalies (1980-1990)
Count
Temperature Anomaly
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

```
Analyzing period: 1990-2000
Number of Outliers: 72
```

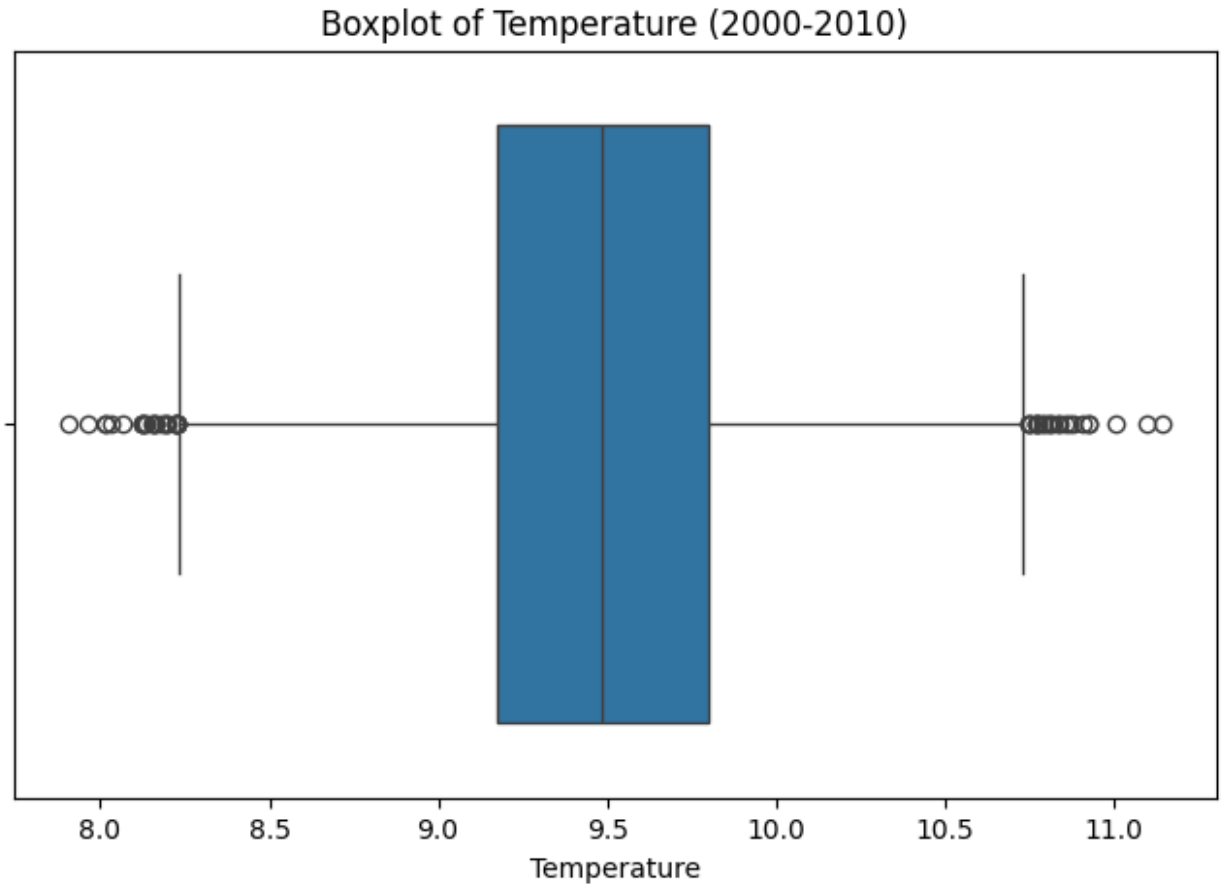


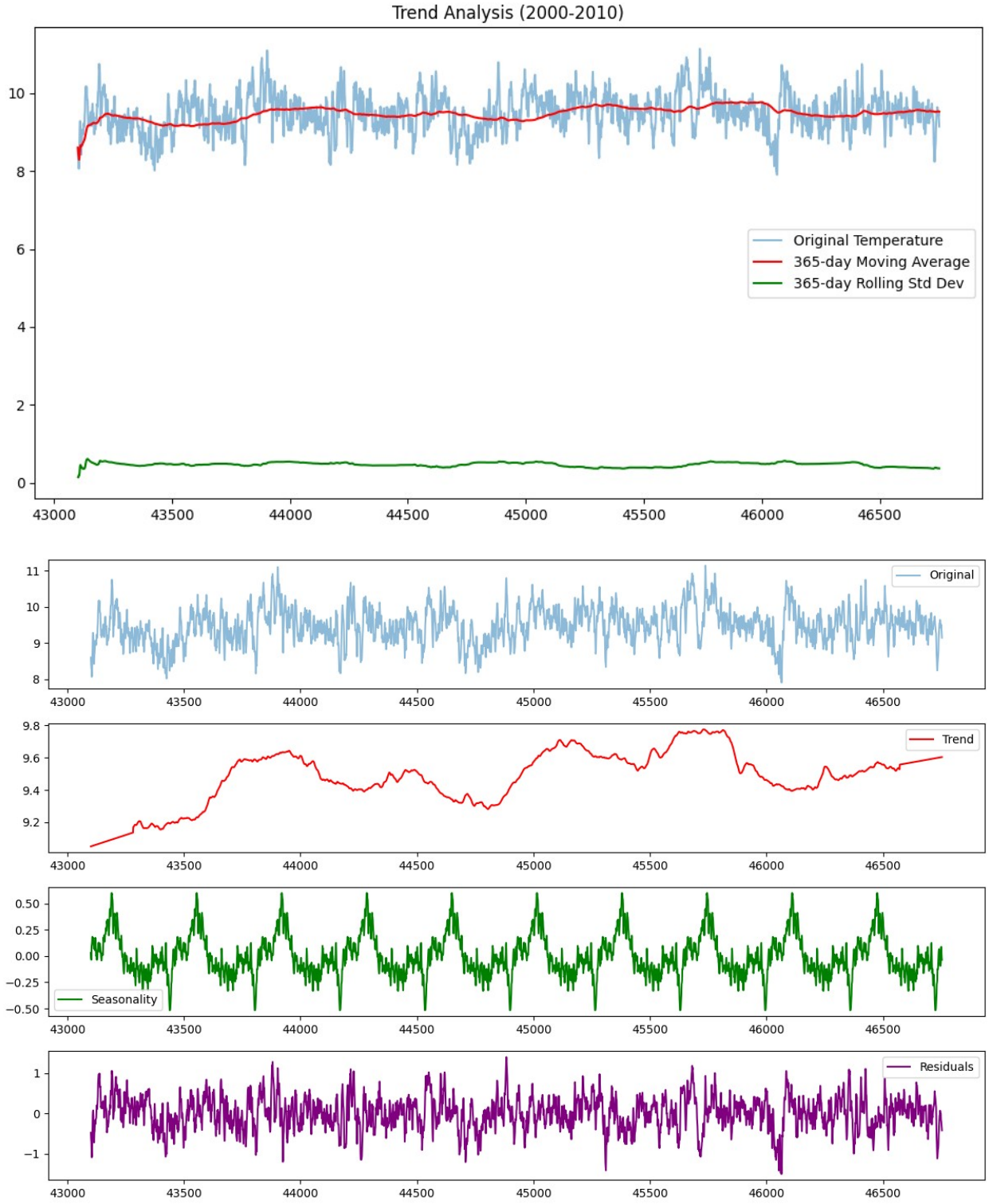


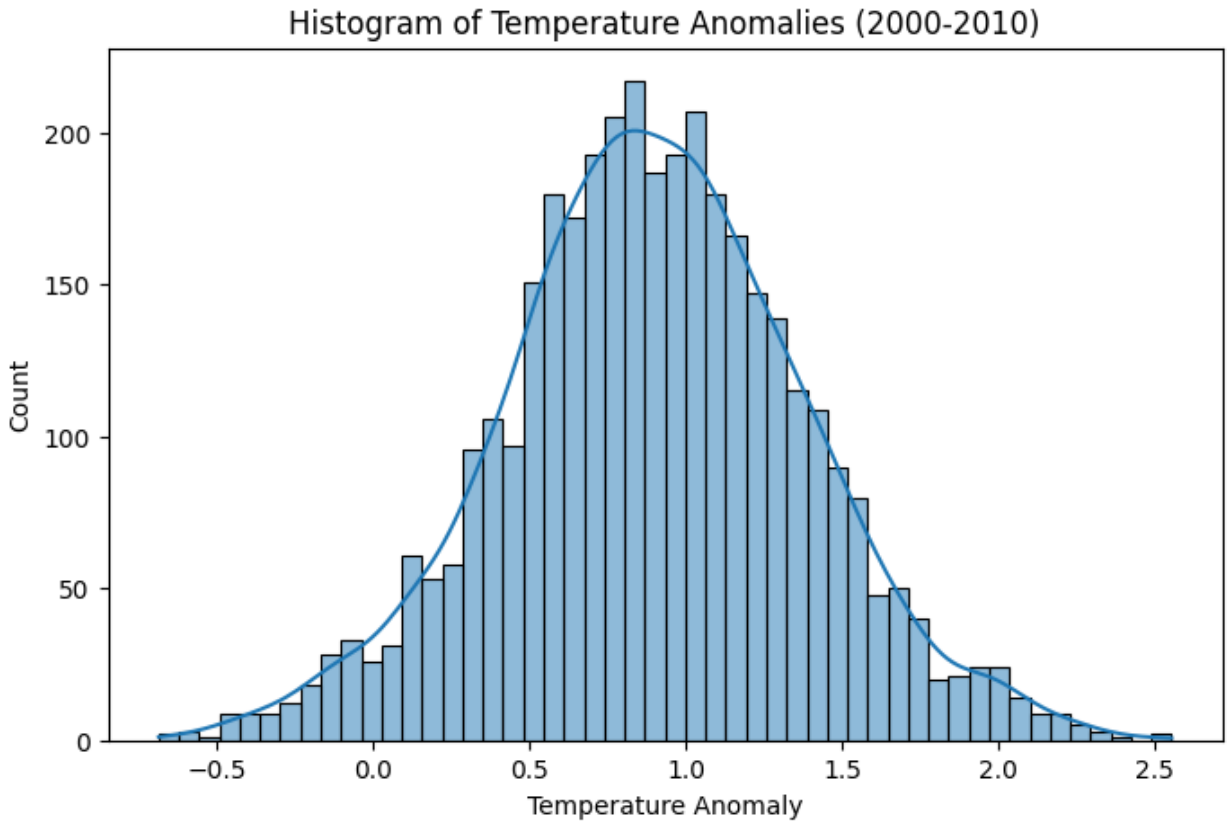

Original
Trend
Seasonality
Residuals
39500
40000
40500
41000
41500
42000
42500
43000


Histogram of Temperature Anomalies (1990-2000)
Count
Temperature Anomaly
0
50
100
150
200
250
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

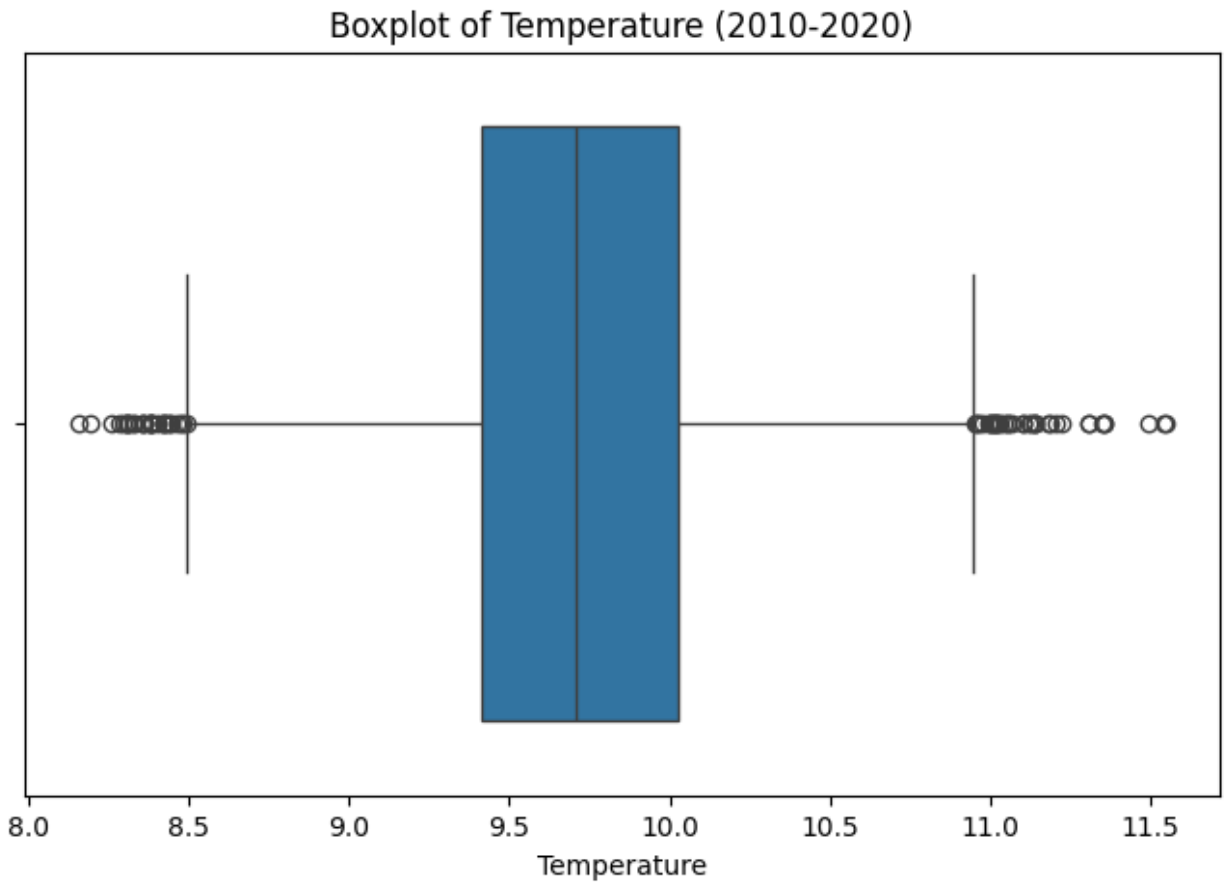
```
Analyzing period: 2000-2010
Number of Outliers: 48
```

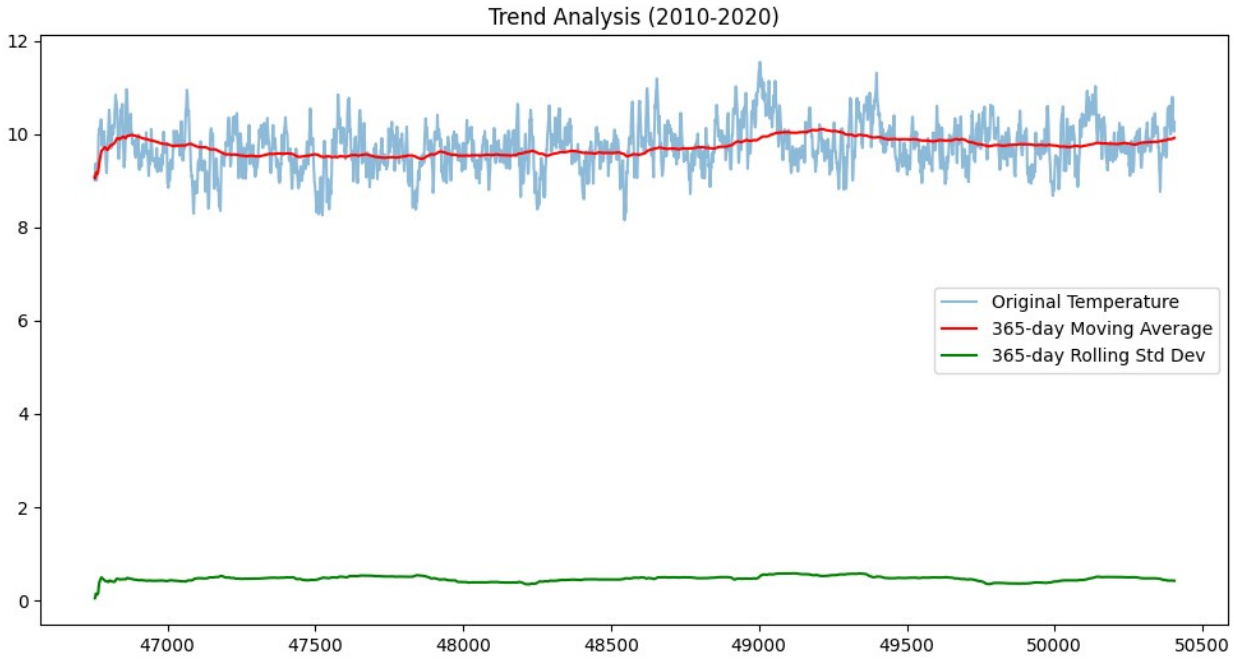


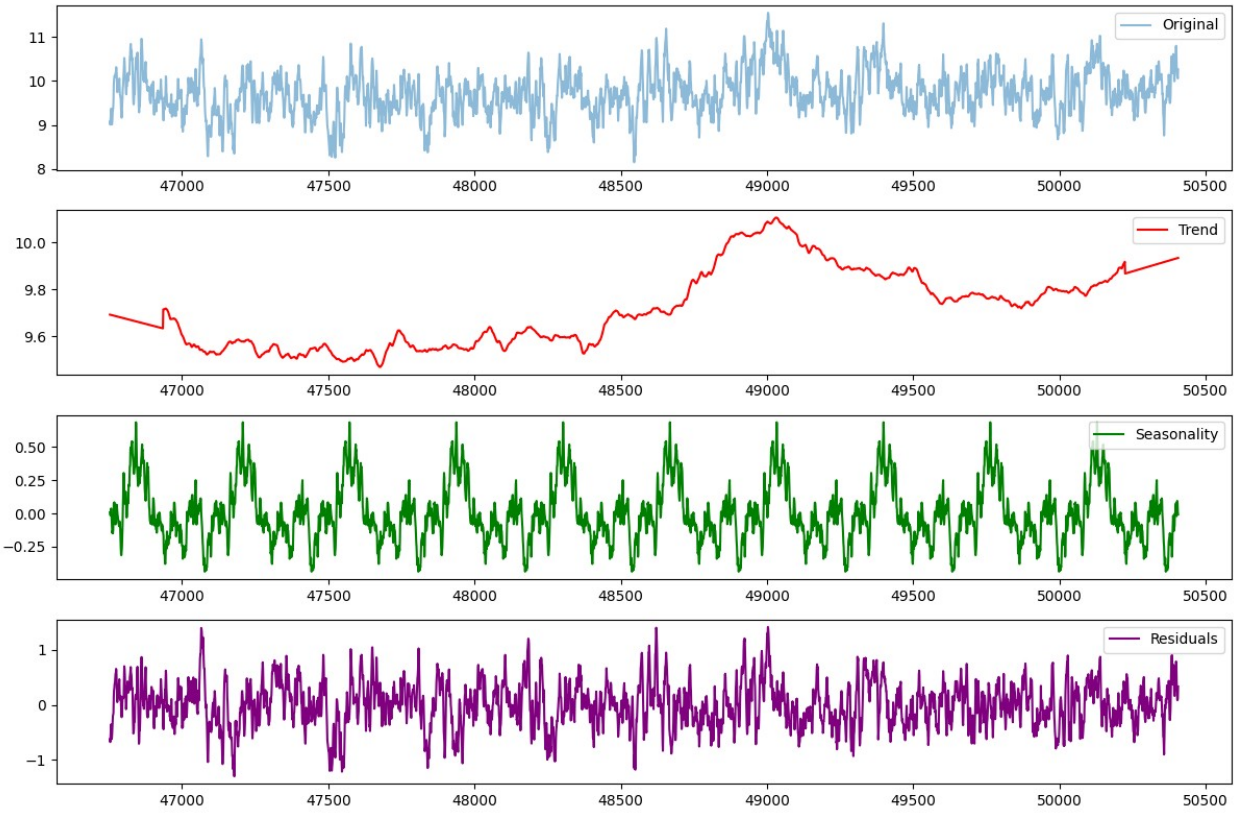

Trend Analysis (2000-2010)
Original Temperature
365-day Moving Average
365-day Rolling Std Dev
Original
Trend
Seasonality
Residuals

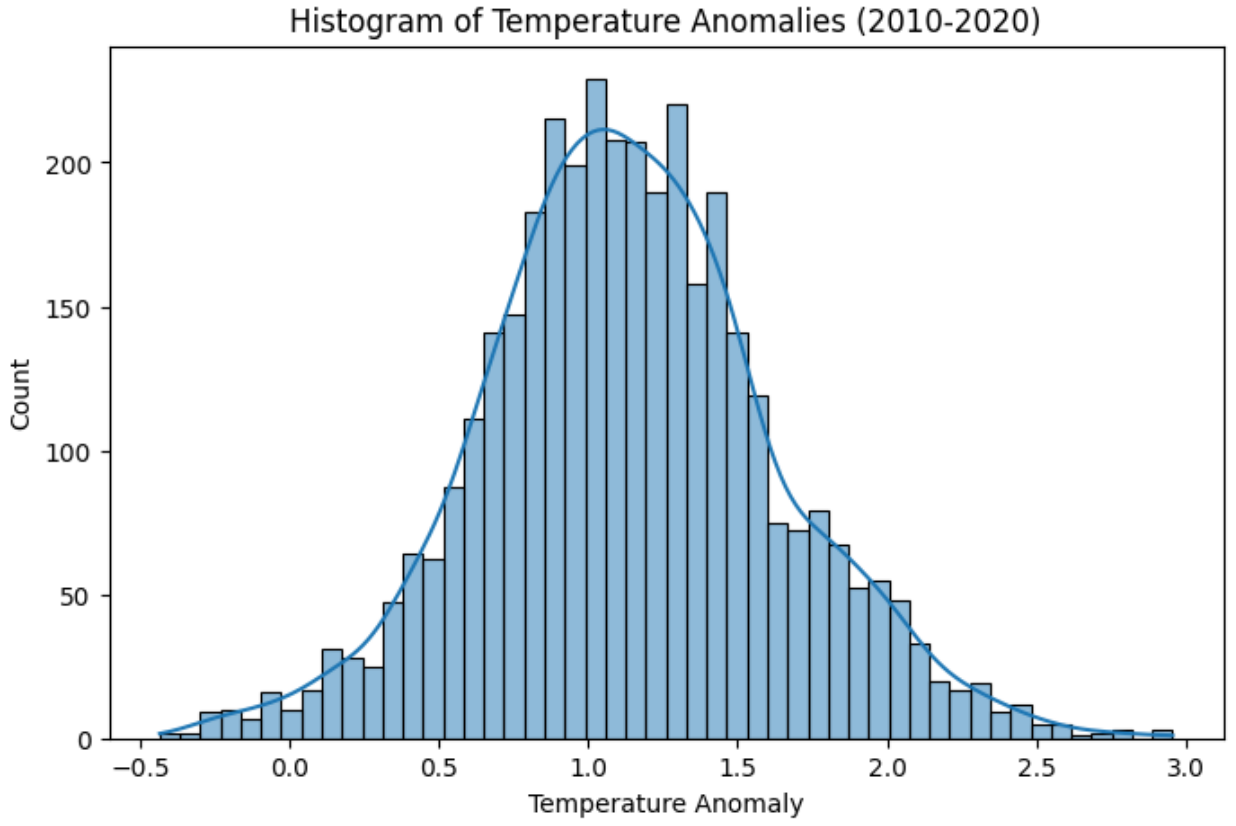



```
Analyzing period: 2010-2020
Number of Outliers: 67
```


Boxplot of Temperature (2010-2020)
8.0
8.5
9.0
9.5
10.0
10.5
11.0
11.5
Temperature


Trend Analysis (2010-2020)
12
10
8
6
4
2
0
47000
47500
48000
48500
49000
49500
50000
50500
Original Temperature
365-day Moving Average
365-day Rolling Std Dev


Original
Trend
Seasonality
Residuals
47000
47500
48000
48500
49000
49500
50000
50500


Histogram of Temperature Anomalies (2010-2020)
Count
Temperature Anomaly
0
50
100
150
200
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0

Mean significance tests

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import ttest_ind, f_oneway, shapiro
import scipy.stats as stats

# Convert 'Date' column to datetime (modify column name if needed)
if 'Date' in df.columns:
    df['Date'] = pd.to_datetime(df['Date'])
    df.set_index('Date', inplace=True)

# Drop missing values
df.dropna(subset=['Temperature'], inplace=True)
print(df.head())
### ---- 1. Hypothesis Testing ---- ###

# Splitting the data into two periods for t-test
early_period = df[df.index.year < 1950]['Temperature']
recent_period = df[df.index.year >= 2000]['Temperature']

# Independent t-test (assumes normality)
t_stat, p_value = ttest_ind(early_period, recent_period,
equal_var=False)
print(f"T-test results: t-statistic={t_stat:.4f}, p-
value={p_value:.4f}")

if p_value < 0.05:
    print("Reject H0: The mean temperature has significantly changed
over time.")
else:
    print("Fail to reject H0: No significant change in mean
temperature.")

# ANOVA test for multiple time periods (dividing data into 50-year
bins)
df['Year'] = df.index.year
df['Period'] = pd.cut(df['Year'], bins=[1880, 1930, 1980, 2020],
labels=['1880-1930', '1930-1980', '1980-2020'])

# Perform ANOVA
anova_groups = [df[df['Period'] == period]['Temperature'] for period
in df['Period'].unique()]
anova_stat, anova_p = f_oneway(*anova_groups)
print(f"ANOVA results: F-statistic={anova_stat:.4f}, p-
value={anova_p:.4f}")

if anova_p < 0.05:
```

```
    print("Reject H0: Mean temperatures differ significantly across
time periods.")
else:
    print("Fail to reject H0: No significant difference in mean
temperatures across periods.")

### ---- 2. Fitting a Multivariate Normal Distribution ---- ###

# Check normality assumption (Shapiro-Wilk Test on a sample)
sample = df['Temperature'].sample(500, random_state=42)  # Take a
sample for normality check
shapiro_stat, shapiro_p = shapiro(sample)
print(f"Shapiro-Wilk Test: Statistic={shapiro_stat:.4f}, p-
value={shapiro_p:.4f}")

if shapiro_p < 0.05:
    print("Data is not normally distributed (p < 0.05). Consider
transformations.")
else:
    print("Data follows normal distribution (p >= 0.05).")

# QQ Plot for normality check
plt.figure(figsize=(6, 6))
stats.probplot(sample, dist="norm", plot=plt)
plt.title("QQ Plot of Temperature")
plt.show()

# Compute Covariance Matrix
cov_matrix = np.cov(df[['Temperature', 'Anomaly']].dropna().T)
print("Covariance Matrix:\n", cov_matrix)
```

```
   Year  Month  Day  Day of Year  Anomaly  Temperature
Rolling_Mean  \
0  1880      1    1            1   -0.692        7.898      7.898000

1  1880      1    2            2   -0.592        7.998      7.948000

2  1880      1    3            3   -0.673        7.917      7.937667

3  1880      1    4            4   -0.615        7.975      7.947000

4  1880      1    5            5   -0.681        7.909      7.939400


   Rolling_Std
0          NaN
1          NaN
2          NaN
3          NaN
4          NaN
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[8], line 19
     15 print(df.head())
     16 ### ---- 1. Hypothesis Testing ---- ###
     17
     18 # Splitting the data into two periods for t-test
---> 19 early_period = df[df.index.year < 1950]['Temperature']
     20 recent_period = df[df.index.year >= 2000]['Temperature']
     22 # Independent t-test (assumes normality)

AttributeError: 'RangeIndex' object has no attribute 'year'
```

10 years group study and 35 years group study

```python
# Extract Year and compute yearly average temperature
df["Year"] = df.index.year
yearly_avg_temp = df.groupby("Year")
["Temperature"].mean().reset_index()

# Define group configurations
group_configs = {
    "10-Year Groups": (14, 10),
    "35-Year Groups": (4, 35),
}

# Dictionary to store all results
all_distribution_stats = {}

# Perform analysis for each group size
for group_name, (num_groups, group_size) in group_configs.items():

    # Create figure for subplots
    fig, axes = plt.subplots(num_groups, 2, figsize=(12, 5 *
num_groups))

    # Dictionary to store statistics for the current group
    distribution_stats = {}

    # Perform distribution analysis for each group
    for i in range(num_groups):
        start_index = i * group_size
        end_index = start_index + group_size

        if end_index > len(yearly_avg_temp):
            break  # Stop if out of range
```

```
        # Extract temperature data for the group
        group_data = yearly_avg_temp.iloc[start_index:end_index]
["Temperature"]
        start_year = yearly_avg_temp.iloc[start_index]["Year"]
        end_year = yearly_avg_temp.iloc[end_index - 1]["Year"]
        group_label = f"{start_year}-{end_year}"

        # Compute statistical metrics
        stats_dict = {
            "Mean": np.mean(group_data),
            "Median": np.median(group_data),
            "Variance": np.var(group_data, ddof=1),
            "Skewness": stats.skew(group_data),
            "Kurtosis": stats.kurtosis(group_data),
        }
        distribution_stats[group_label] = stats_dict

        # Plot Histogram & KDE
        sns.histplot(group_data, bins=10, kde=True, ax=axes[i, 0],
color="skyblue")
        axes[i, 0].set_title(f"Histogram & KDE: {group_label}")
        axes[i, 0].set_xlabel("Temperature")

        # Q-Q Plot (Normality Check)
        stats.probplot(group_data, dist="norm", plot=axes[i, 1])
        axes[i, 1].set_title(f"Q-Q Plot: {group_label}")

    # Convert dictionary to DataFrame and store results
    distribution_df = pd.DataFrame.from_dict(distribution_stats,
orient="index")
    all_distribution_stats[group_name] = distribution_df

    # Save statistics to CSV
    filename = f"temperature_distribution_{group_name.replace(' ',
'_').lower()}.csv"
    distribution_df.to_csv(filename)

    # Show plots
    plt.tight_layout()
    plt.show()

    # Print DataFrame
    print(f"□ Distribution Statistics for {group_name}:")
    print(distribution_df)
    print("\n" + "=" * 80 + "\n")

df["Year"] = df.index.year
yearly_avg_temp = df.groupby("Year")
["Temperature"].mean().reset_index()
```

```
# Define the number of groups and group size
num_groups = 14
group_size = 10

# Plot KDE curves for each group
plt.figure(figsize=(12, 6))

for i in range(num_groups):
    start_index = i * group_size
    end_index = start_index + group_size

    if end_index > len(yearly_avg_temp):
        break  # Stop if out of range

    # Extract temperature data for the group
    group_data = yearly_avg_temp.iloc[start_index:end_index]
["Temperature"]
    start_year = yearly_avg_temp.iloc[start_index]["Year"]
    end_year = yearly_avg_temp.iloc[end_index - 1]["Year"]

    # Plot KDE for the group
    sns.kdeplot(group_data, label=f"{start_year}-{end_year}",
linewidth=2)

# Add labels and title
plt.xlabel("Temperature", fontsize=12)
plt.ylabel("Density", fontsize=12)
plt.title("Kernel Density Estimation (KDE) of Temperature
Distributions (14 Groups of 10 Years)", fontsize=14)
plt.legend(title="Year Ranges", bbox_to_anchor=(1.05, 1), loc='upper
left')
plt.grid(True)

# Show plot
plt.show()
```

20 years group study

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats

# Extract Year, Month, and Day
df["Year"] = df.index.year
df["Month"] = df.index.month
```

```
df["Day"] = df.index.day


# Define the number of groups and group size
num_groups = 7
group_size = 20

# Create an empty dictionary to store statistics
distribution_stats = {}

# Create plots
fig, axes = plt.subplots(num_groups, 3, figsize=(18, 5 * num_groups))
# Added an extra column for log-normality

# Perform distribution analysis for each group
for i in range(num_groups):
    start_index = i * group_size
    end_index = start_index + group_size

    if end_index > len(df):
        break  # Stop if out of range

    # Extract daily temperature data for the group
    group_data = df.iloc[start_index:end_index]["Temperature"]
    group_label = f"Group {i+1} ({df.iloc[start_index]['Year']}-
{df.iloc[start_index]['Month']}-{df.iloc[start_index]['Day']} to
{df.iloc[end_index-1]['Year']}-{df.iloc[end_index-1]['Month']}-
{df.iloc[end_index-1]['Day']})"

    # Compute descriptive statistics
    stats_dict = {
        "Mean": np.mean(group_data),
        "Median": np.median(group_data),
        "Variance": np.var(group_data, ddof=1),
        "Skewness": stats.skew(group_data),
        "Kurtosis": stats.kurtosis(group_data),
    }

    # **Normality Test** (Shapiro-Wilk)
    shapiro_test = stats.shapiro(group_data)
    stats_dict["Normality (Shapiro-Wilk) p-value"] =
shapiro_test.pvalue

    # **Log-Normality Test** (Kolmogorov-Smirnov Test)
    log_data = np.log(group_data[group_data > 0])  # Take log, avoid
log(0) issues
    ks_test = stats.kstest(log_data, 'norm', args=(np.mean(log_data),
np.std(log_data)))
    stats_dict["Log-Normality (KS test) p-value"] = ks_test.pvalue
```

```
    distribution_stats[group_label] = stats_dict

    # **Histogram & KDE plot**
    sns.histplot(group_data, bins=10, kde=True, ax=axes[i, 0],
color="skyblue")
    axes[i, 0].set_title(f"Histogram & KDE: {group_label}")
    axes[i, 0].set_xlabel("Temperature")

    # **Q-Q Plot for Normality**
    stats.probplot(group_data, dist="norm", plot=axes[i, 1])
    axes[i, 1].set_title(f"Q-Q Plot: {group_label}")

    # **Q-Q Plot for Log-Normality**
    stats.probplot(log_data, dist="norm", plot=axes[i, 2])
    axes[i, 2].set_title(f"Log-Normal Q-Q Plot: {group_label}")

# Save distribution statistics to a DataFrame
distribution_df = pd.DataFrame.from_dict(distribution_stats,
orient="index")

# Show plots
plt.tight_layout()
plt.show()

# Save statistics to CSV
distribution_df.to_csv("daily_temperature_distribution_study_with_norm
ality.csv")

# Print results
print(distribution_df)

# Define the number of groups and group size
num_groups = 7
group_size = 20

# Plot KDE curves for each group
plt.figure(figsize=(12, 6))

for i in range(num_groups):
    start_index = i * group_size
    end_index = start_index + group_size

    if end_index > len(yearly_avg_temp):
        break  # Stop if out of range

    # Extract temperature data for the group
    group_data = yearly_avg_temp.iloc[start_index:end_index]
["Temperature"]
    start_year = yearly_avg_temp.iloc[start_index]["Year"]
    end_year = yearly_avg_temp.iloc[end_index - 1]["Year"]
```

```
    # Plot KDE for the group
    sns.kdeplot(group_data, label=f"{start_year}-{end_year}",
linewidth=2)

# Add labels and title
plt.xlabel("Temperature", fontsize=12)
plt.ylabel("Density", fontsize=12)
plt.title("Kernel Density Estimation (KDE) of Temperature
Distributions (7 Groups of 20 Years)", fontsize=14)
plt.legend(title="Year Ranges")
plt.grid(True)

# Show plot
plt.show()
```

Growth Rate calculations

Annual growth rate and n- yearly growth rates (n=10,20,35,70)

```
import pandas as pd
import numpy as np

# Load data (Assuming df is already loaded and indexed by Date)

# Extract Year and calculate yearly average temperature
df["Year"] = df.index.year
yearly_avg_temp = df.groupby("Year")
["Temperature"].mean().reset_index()

# Function to calculate CAGR using two formulas
def calculate_cagr1(initial, final, years):
    return ((final / initial) ** (1 / years)) - 1 if years > 0 else
np.nan

def calculate_cagr2(initial, final, years):
    return ((final / initial)) - 1 if years > 0 else np.nan

# Define the group sizes and names
group_sizes = [10, 20, 35, 70]  # Groups of 10, 20, 35, and 70 years
group_labels = ["10-Year Growth", "20-Year Growth", "35-Year Growth",
"70-Year Growth"]

all_growth_data = {}
```

```python
# Loop through each group size and compute CAGR using both formulas
for group_size, label in zip(group_sizes, group_labels):
    num_groups = len(yearly_avg_temp) // group_size  # Number of full
groups
    growth_list1, growth_list2 = [], []

    for i in range(num_groups):
        start_index = i * group_size
        end_index = start_index + group_size - 1

        if end_index < len(yearly_avg_temp):
            temp_initial = yearly_avg_temp.iloc[start_index]
["Temperature"]
            temp_final = yearly_avg_temp.iloc[end_index]
["Temperature"]
            growth_list1.append(calculate_cagr1(temp_initial,
temp_final, group_size))
            growth_list2.append(calculate_cagr2(temp_initial,
temp_final, group_size))
        else:
            break  # Stop if the range goes out of bounds

    # Store results
    all_growth_data[f"{label} - CAGR1"] = growth_list1
    all_growth_data[f"{label} - CAGR2"] = growth_list2

# Find the maximum length among the lists
max_length = max(map(len, all_growth_data.values()))

# Pad shorter lists with NaN to match the maximum length
for key in all_growth_data:
    all_growth_data[key] += [np.nan] * (max_length -
len(all_growth_data[key]))

# Convert to DataFrame
growth_df = pd.DataFrame.from_dict(all_growth_data)

growth_df.index = [f"Group {i+1}" for i in range(len(growth_df))]

# Save to CSV
growth_df.to_csv("temperature1_growth_analysis.csv")

# Display results
print(growth_df)
```

Time Series Modelling

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_absolute_error, mean_squared_error
from pmdarima import auto_arima

# Exploratory Analysis
plt.figure(figsize=(14,6))
plt.plot(df['Temperature'], label='Daily Temperature')
plt.title('Daily Global Temperature Over Time')
plt.xlabel('Index')
plt.ylabel('Temperature')
plt.legend()
plt.savefig("daily_global_temperature.png")  # Save with a meaningful
filename
plt.close()

# ADF Test for Stationarity
def adf_test(series):
    result = adfuller(series.dropna())
    print(f"ADF Statistic: {result[0]}")
    print(f"p-value: {result[1]}")
    if result[1] > 0.05:
        print("Non-stationary series (p > 0.05)")
    else:
        print("Stationary series (p <= 0.05)")

print("ADF Test (Original Series):")
adf_test(df['Temperature'])

# Differencing
df['Temperature_Diff'] = df['Temperature'].diff()

print("\nADF Test (After Differencing):")
adf_test(df['Temperature_Diff'])

# ACF and PACF
fig, axes = plt.subplots(1, 2, figsize=(14,5))
plot_acf(df['Temperature_Diff'].dropna(), ax=axes[0])
plot_pacf(df['Temperature_Diff'].dropna(), ax=axes[1])
plt.savefig("acf_pacf_analysis.png")  # Save with an appropriate
filename
plt.close()
```

```
# Automatically select best ARIMA order
auto_model = auto_arima(
    df['Temperature'],
    seasonal=False,
    stepwise=True,
    suppress_warnings=True,
    error_action='ignore',
    trace=True,
    max_p=6,
    max_q=6
)

print(f"\nBest ARIMA Order Found: {auto_model.order}")

# Fit ARIMA Model
model = ARIMA(df['Temperature'], order=auto_model.order)
arima_result = model.fit()
print(arima_result.summary())

# Forecast next 365 days
forecast = arima_result.forecast(steps=365)

# Plot
plt.figure(figsize=(14,6))
plt.plot(df['Temperature'][-365:], label='Actual')
plt.plot(np.arange(len(df), len(df) + 365), forecast,
label='Forecast', linestyle='dashed')
plt.title('ARIMA Forecast vs Actual (Last 365 Days)')
plt.xlabel('Index')
plt.ylabel('Temperature')
plt.legend()
plt.savefig("arima_forecast_vs_actual.png")  # Save with an
appropriate filename
plt.close()


# Evaluation
y_actual = df['Temperature'][-365:]
y_pred = forecast[:365]
mae = mean_absolute_error(y_actual, y_pred)
rmse = np.sqrt(mean_squared_error(y_actual, y_pred))
print(f"\nARIMA Performance on Full Dataset:")
print(f"MAE: {mae:.4f}")
print(f"RMSE: {rmse:.4f}")

ADF Test (Original Series):
ADF Statistic: -9.437660805688283
p-value: 4.991882507815218e-16
Stationary series (p <= 0.05)
```

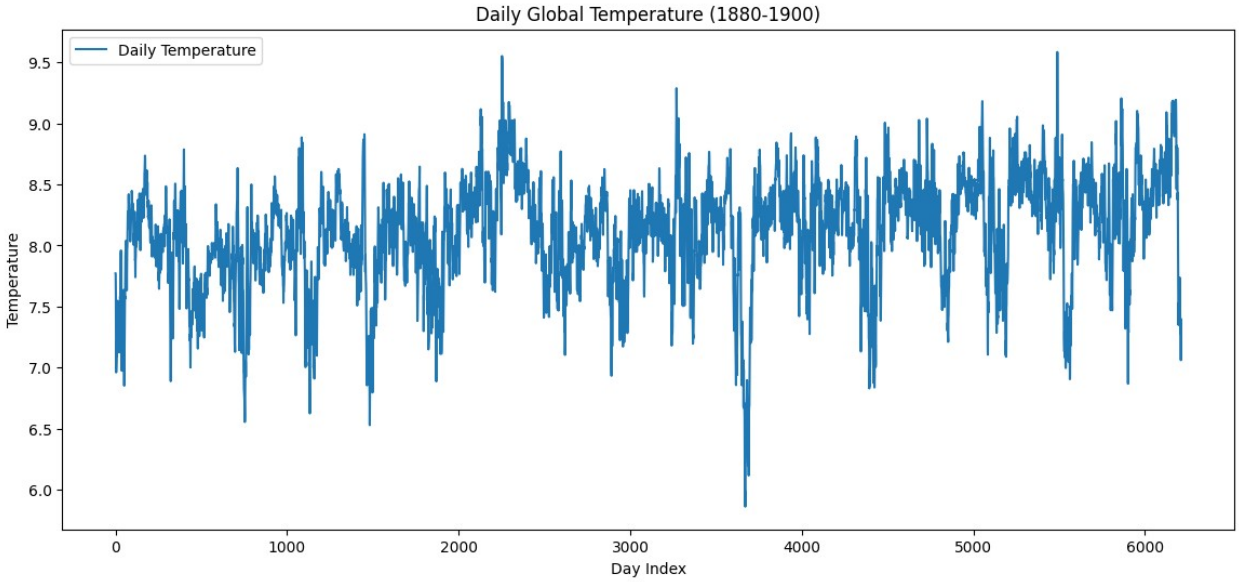
```
ADF Test (After Differencing):
ADF Statistic: -43.762296113685935
p-value: 0.0
Stationary series (p <= 0.05)
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=-57163.089, Time=29.44 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-48399.827, Time=2.20 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-53767.450, Time=1.42 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-54586.133, Time=3.08 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=-48401.822, Time=1.34 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=-54582.434, Time=8.30 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=-56872.395, Time=29.94 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=inf, Time=33.32 sec
 ARIMA(2,1,3)(0,0,0)[0] intercept   : AIC=inf, Time=38.43 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=-54594.783, Time=3.82 sec
 ARIMA(1,1,3)(0,0,0)[0] intercept   : AIC=-57204.109, Time=32.06 sec
 ARIMA(0,1,3)(0,0,0)[0] intercept   : AIC=-54821.407, Time=8.26 sec
 ARIMA(1,1,4)(0,0,0)[0] intercept   : AIC=-57172.578, Time=39.37 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=-54599.336, Time=6.09 sec
 ARIMA(0,1,4)(0,0,0)[0] intercept   : AIC=-55238.337, Time=18.25 sec
 ARIMA(2,1,4)(0,0,0)[0] intercept   : AIC=-57161.968, Time=45.42 sec
 ARIMA(1,1,3)(0,0,0)[0]             : AIC=-57205.935, Time=13.19 sec
 ARIMA(0,1,3)(0,0,0)[0]             : AIC=-54823.403, Time=4.57 sec
 ARIMA(1,1,2)(0,0,0)[0]             : AIC=-54632.880, Time=10.79 sec
 ARIMA(2,1,3)(0,0,0)[0]             : AIC=-57115.991, Time=21.29 sec
 ARIMA(1,1,4)(0,0,0)[0]             : AIC=-57206.792, Time=15.39 sec
 ARIMA(0,1,4)(0,0,0)[0]             : AIC=-55240.332, Time=4.13 sec
 ARIMA(2,1,4)(0,0,0)[0]             : AIC=-57204.926, Time=15.86 sec
 ARIMA(1,1,5)(0,0,0)[0]             : AIC=inf, Time=18.30 sec
 ARIMA(0,1,5)(0,0,0)[0]             : AIC=-55650.498, Time=5.85 sec
 ARIMA(2,1,5)(0,0,0)[0]             : AIC=-57202.826, Time=25.41 sec

Best model:  ARIMA(1,1,4)(0,0,0)[0]
Total fit time: 435.555 seconds

Best ARIMA Order Found: (1, 1, 4)
                               SARIMAX Results

==============================================================================
========
Dep. Variable:            Temperature   No. Observations:
50985
Model:                 ARIMA(1, 1, 4)   Log Likelihood
28609.396
Date:                Tue, 08 Apr 2025   AIC                            -
57206.792
Time:                        15:58:53   BIC                            -
57153.756
Sample:                             0   HQIC                           -
57190.194
```

```
                                 - 50985

Covariance Type:                   opg

==============================================================================
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
ar.L1          0.8650      0.003    256.401      0.000       0.858
0.872
ma.L1         -0.5537      0.005   -105.126      0.000      -0.564
-0.543
ma.L2         -0.3358      0.005    -69.416      0.000      -0.345
-0.326
ma.L3         -0.0832      0.005    -16.923      0.000      -0.093
-0.074
ma.L4         -0.0092      0.005     -1.920      0.055      -0.019
0.000
sigma2         0.0191      0.000    175.469      0.000       0.019
0.019
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
374.89
Prob(Q):                              0.98   Prob(JB):
0.00
Heteroskedasticity (H):               1.07   Skew:
-0.03
Prob(H) (two-sided):                  0.00   Kurtosis:
3.42
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).

ARIMA Performance on Full Dataset:
MAE: 0.3041
RMSE: 0.3773

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

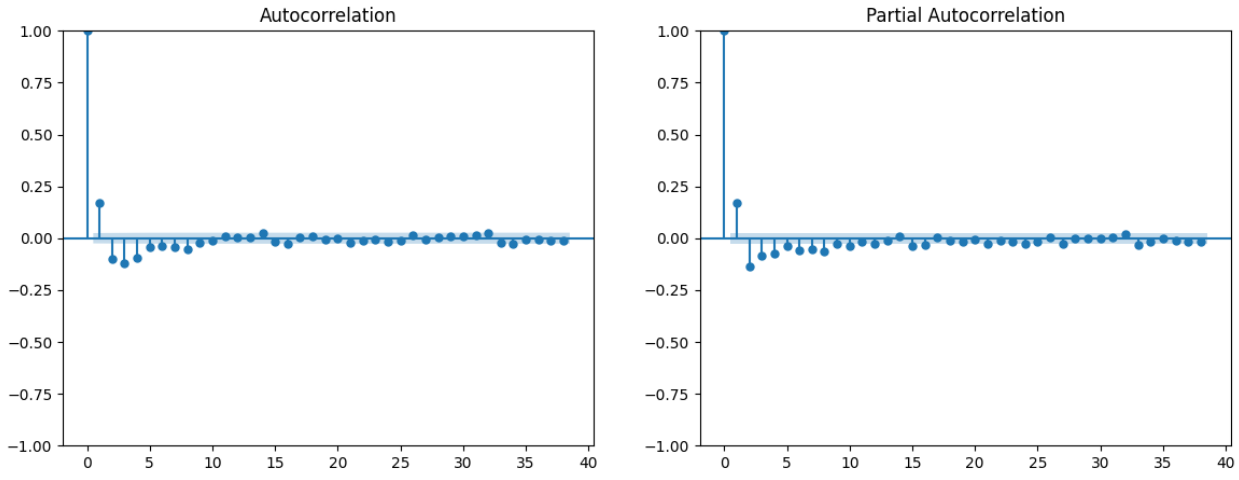
```
from pmdarima import auto_arima

group_size = 20
start_year = 1880
end_year = 2020

def adf_test(series):
    result = adfuller(series.dropna())
    print(f"ADF Statistic: {result[0]}")
    print(f"p-value: {result[1]}")
    if result[1] > 0.05:
        print("Non-stationary series (p > 0.05)")
    else:
        print("Stationary series (p <= 0.05)")

results_list = []

for start in range(start_year, end_year, group_size):
    df_group = df[(df['Year'] >= start) & (df['Year'] < start +
group_size)].copy()
    df_group.reset_index(drop=True, inplace=True)

    print(f"\nAnalyzing period: {start}-{start + group_size}")

    # Plot original temperature
    plt.figure(figsize=(14,6))
    plt.plot(df_group['Temperature'], label='Daily Temperature')
    plt.title(f'Daily Global Temperature ({start}-{start +
group_size})')
    plt.xlabel('Day Index')
    plt.ylabel('Temperature')
    plt.legend()
    plt.show()

    # ADF Test
    adf_test(df_group['Temperature'])

    # Differencing
    df_group['Temperature_Diff'] = df_group['Temperature'].diff()
    adf_test(df_group['Temperature_Diff'].dropna())

    # ACF and PACF
    fig, axes = plt.subplots(1, 2, figsize=(14,5))
    plot_acf(df_group['Temperature_Diff'].dropna(), ax=axes[0])
    plot_pacf(df_group['Temperature_Diff'].dropna(), ax=axes[1])
    plt.show()

    # Auto ARIMA Order Selection
    auto_model = auto_arima(
        df_group['Temperature'],
```

```
        seasonal=False,
        stepwise=True,
        suppress_warnings=True,
        error_action='ignore',
        trace=True,
        max_p=6,
        max_q=6
    )

    print(f"Selected Order (p,d,q): {auto_model.order}")

    # Fit ARIMA using selected order
    model = ARIMA(df_group['Temperature'], order=auto_model.order)
    model_fit = model.fit()
    print(model_fit.summary())

    # Forecast next 4 values
    forecast = model_fit.forecast(steps=4)

    # Plot actual and forecast
    plt.figure(figsize=(14,6))
    plt.plot(df_group['Temperature'][-10:], label='Actual')
    plt.plot(
        np.arange(len(df_group), len(df_group)+4),
        forecast,
        label='Forecast',
        linestyle='dashed'
    )
    plt.title(f'ARIMA Forecast vs Actual ({start}-{start +
group_size})')
    plt.legend()
    plt.show()

    # Performance metrics using actual last 4 values
    y_actual = df_group['Temperature'][-4:].values
    y_pred = forecast[:4]
    mae = mean_absolute_error(y_actual, y_pred)
    rmse = np.sqrt(mean_squared_error(y_actual, y_pred))
    print(f'ARIMA MAE ({start}-{start + group_size}): {mae:.4f}, RMSE:
{rmse:.4f}')

    results_list.append({
        "Time Period": f"{start}-{start + group_size}",
        "Order": auto_model.order,
        "MAE": mae,
        "RMSE": rmse
    })

# Final DataFrame of results
results_df = pd.DataFrame(results_list)
```

```
print("\nARIMA Performance Summary:")
print(results_df)


Analyzing period: 1880-1900
```



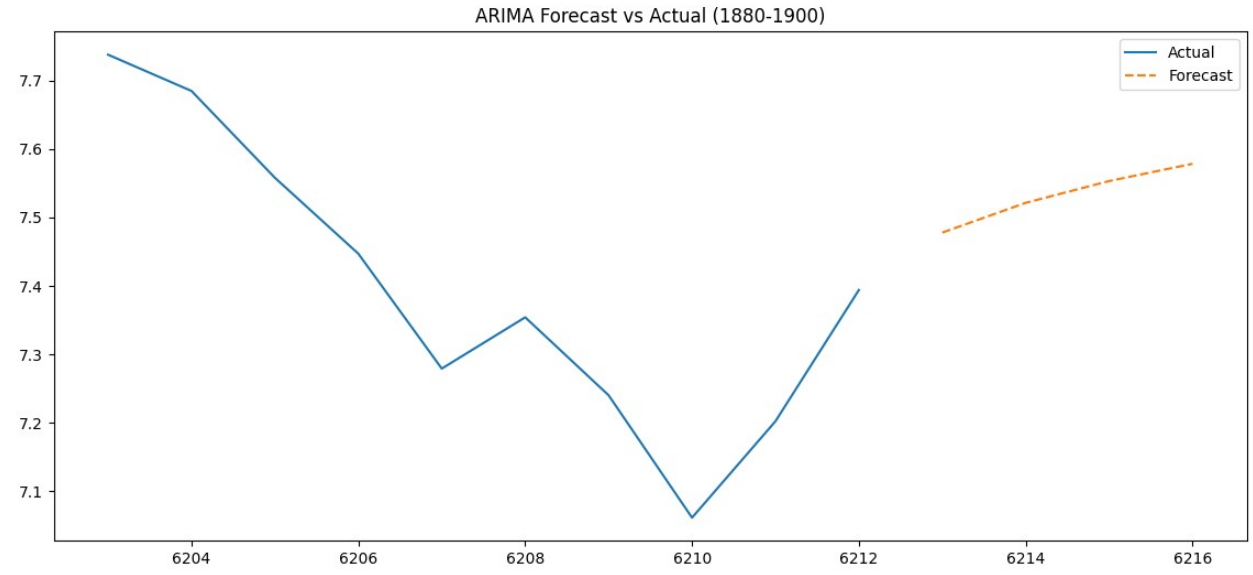

```
ADF Statistic: -9.505999565937447
p-value: 3.3461405645231796e-16
Stationary series (p <= 0.05)
ADF Statistic: -24.730374567814753
p-value: 0.0
Stationary series (p <= 0.05)
```



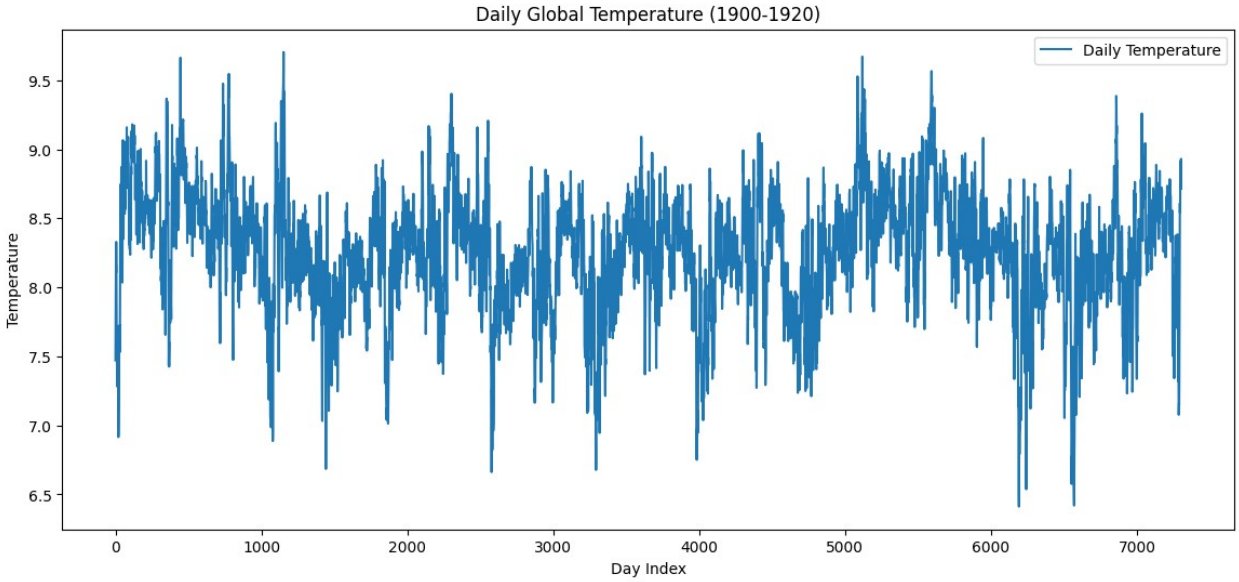

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=-7356.225, Time=4.08 sec
```

```
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-6890.814, Time=0.34 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-7074.713, Time=0.51 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-7120.486, Time=0.57 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=-6892.813, Time=0.16 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=-7360.900, Time=3.05 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=-7150.586, Time=1.63 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=-7134.081, Time=1.28 sec
 ARIMA(1,1,3)(0,0,0)[0] intercept   : AIC=-7362.650, Time=4.74 sec
 ARIMA(0,1,3)(0,0,0)[0] intercept   : AIC=-7222.532, Time=1.04 sec
 ARIMA(2,1,3)(0,0,0)[0] intercept   : AIC=-7357.826, Time=2.19 sec
 ARIMA(1,1,4)(0,0,0)[0] intercept   : AIC=-7361.274, Time=4.74 sec
 ARIMA(0,1,4)(0,0,0)[0] intercept   : AIC=-7295.500, Time=1.42 sec
 ARIMA(2,1,4)(0,0,0)[0] intercept   : AIC=-7359.212, Time=5.32 sec
 ARIMA(1,1,3)(0,0,0)[0]             : AIC=-7364.650, Time=1.21 sec
 ARIMA(0,1,3)(0,0,0)[0]             : AIC=-7224.531, Time=0.35 sec
 ARIMA(1,1,2)(0,0,0)[0]             : AIC=-7363.973, Time=0.88 sec
 ARIMA(2,1,3)(0,0,0)[0]             : AIC=-7362.857, Time=2.27 sec
 ARIMA(1,1,4)(0,0,0)[0]             : AIC=inf, Time=2.42 sec
 ARIMA(0,1,2)(0,0,0)[0]             : AIC=-7152.585, Time=0.31 sec
 ARIMA(0,1,4)(0,0,0)[0]             : AIC=-7297.499, Time=0.48 sec
 ARIMA(2,1,2)(0,0,0)[0]             : AIC=-7364.120, Time=1.69 sec
 ARIMA(2,1,4)(0,0,0)[0]             : AIC=-7362.469, Time=4.57 sec

Best model:  ARIMA(1,1,3)(0,0,0)[0]
Total fit time: 45.300 seconds
Selected Order (p,d,q): (1, 1, 3)
                               SARIMAX Results

==============================================================================
========
Dep. Variable:             Temperature   No. Observations:
6213
Model:                  ARIMA(1, 1, 3)   Log Likelihood
3687.325
Date:                 Tue, 08 Apr 2025   AIC                              -
7364.650
Time:                         16:14:27   BIC                              -
7330.979
Sample:                              0   HQIC                             -
7352.978
                                - 6213

Covariance Type:                   opg

==============================================================================
========
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
--------
```

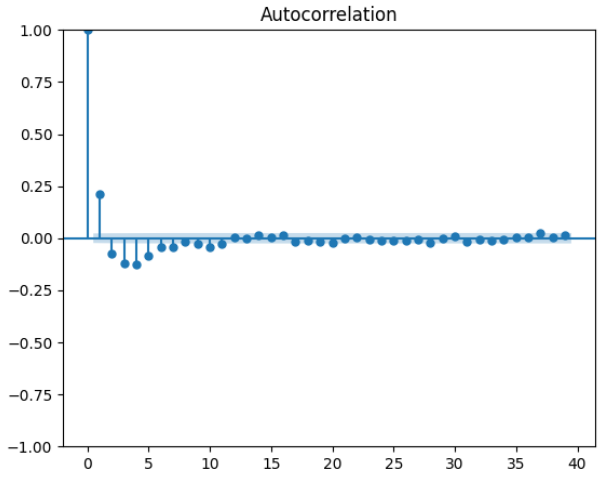
```
ar.L1          0.7916      0.018     43.431      0.000       0.756
0.827
ma.L1         -0.6332      0.021    -30.125      0.000      -0.674
-0.592
ma.L2         -0.2433      0.013    -18.590      0.000      -0.269
-0.218
ma.L3         -0.0317      0.014     -2.318      0.020      -0.059
-0.005
sigma2         0.0179      0.000     65.123      0.000       0.017
0.018
===================================================================
=============
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
144.16
Prob(Q):                              0.99   Prob(JB):
0.00
Heteroskedasticity (H):               1.38   Skew:
-0.06
Prob(H) (two-sided):                  0.00   Kurtosis:
3.74
===================================================================
=============

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
```

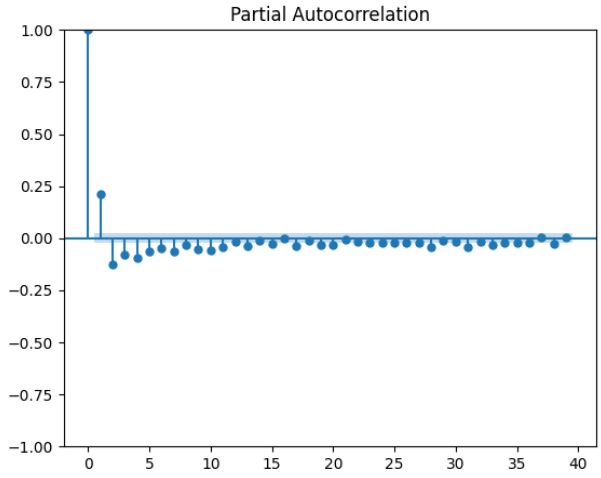



```
ARIMA MAE (1880-1900): 0.3085, RMSE: 0.3264

Analyzing period: 1900-1920
```




```
ADF Statistic: -11.319604391380265
p-value: 1.1830963854949672e-20
Stationary series (p <= 0.05)
ADF Statistic: -21.167372868090293
p-value: 0.0
Stationary series (p <= 0.05)
```

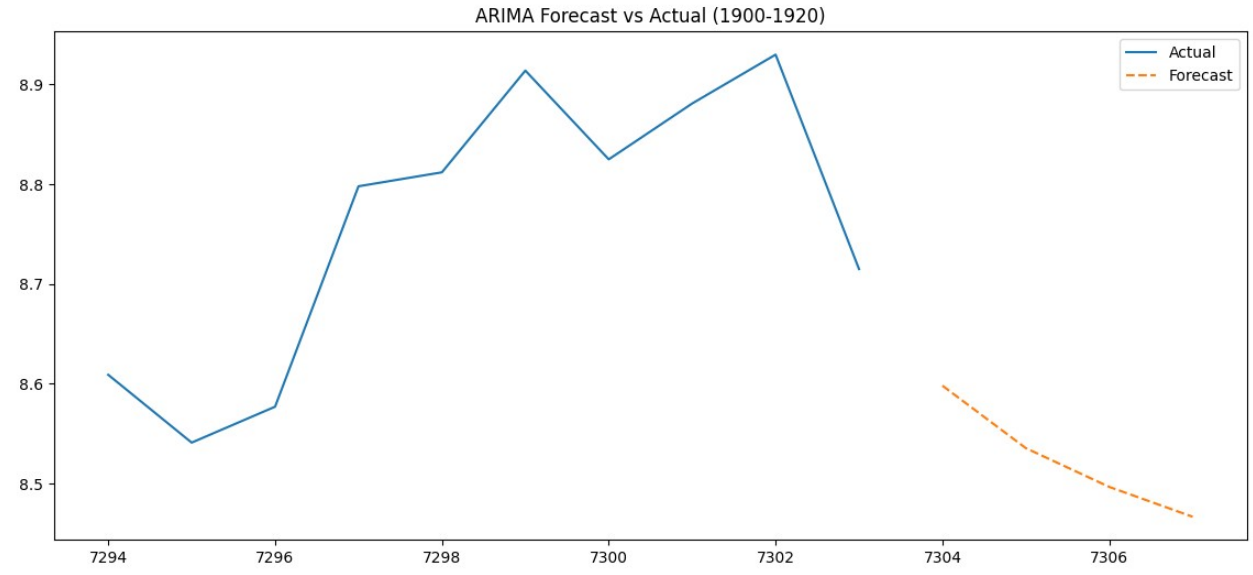




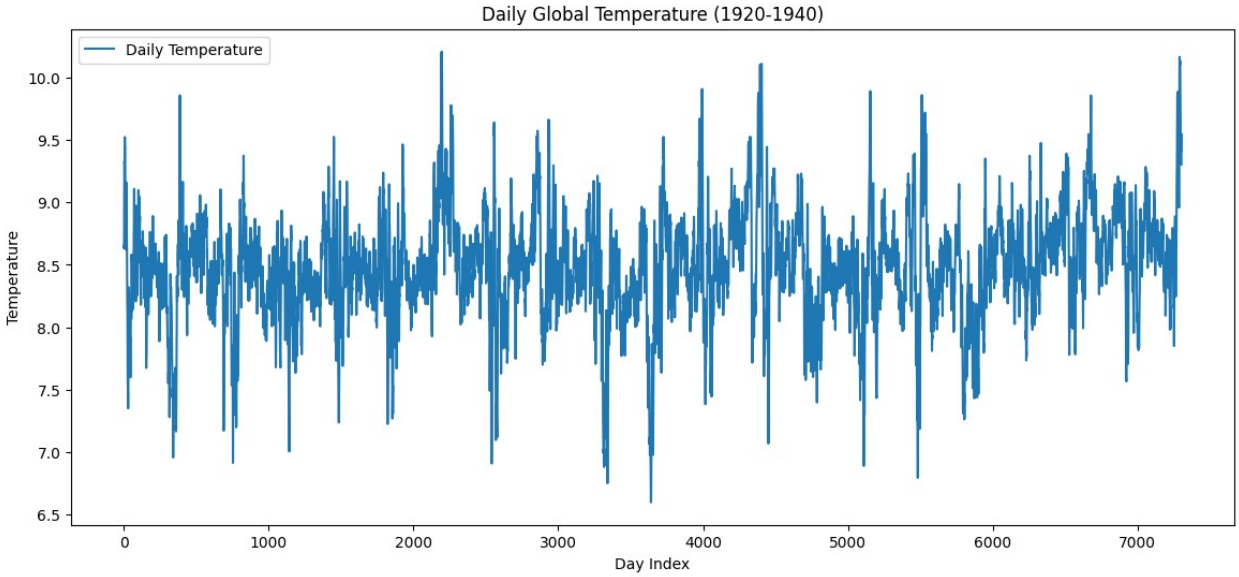

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=-7798.864, Time=4.07 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-7008.392, Time=0.44 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-7349.110, Time=0.31 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-7415.822, Time=0.58 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=-7010.382, Time=0.15 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=inf, Time=5.81 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=-7789.314, Time=5.10 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=-7791.952, Time=4.65 sec
```

```
 ARIMA(2,1,3)(0,0,0)[0] intercept   : AIC=-7789.912, Time=4.96 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=-7420.823, Time=1.64 sec
 ARIMA(1,1,3)(0,0,0)[0] intercept   : AIC=-7794.653, Time=4.56 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=-7804.966, Time=5.13 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=-7511.321, Time=0.74 sec
 ARIMA(4,1,1)(0,0,0)[0] intercept   : AIC=-7804.548, Time=5.47 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=-7466.929, Time=0.35 sec
 ARIMA(4,1,0)(0,0,0)[0] intercept   : AIC=-7576.185, Time=0.99 sec
 ARIMA(4,1,2)(0,0,0)[0] intercept   : AIC=-7799.927, Time=6.10 sec
 ARIMA(3,1,1)(0,0,0)[0]             : AIC=-7806.970, Time=1.35 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=-7791.304, Time=1.14 sec
 ARIMA(3,1,0)(0,0,0)[0]             : AIC=-7513.312, Time=0.40 sec
 ARIMA(4,1,1)(0,0,0)[0]             : AIC=-7806.861, Time=1.75 sec
 ARIMA(3,1,2)(0,0,0)[0]             : AIC=-7811.460, Time=2.74 sec
 ARIMA(2,1,2)(0,0,0)[0]             : AIC=-7805.344, Time=2.16 sec
 ARIMA(4,1,2)(0,0,0)[0]             : AIC=-7803.154, Time=2.85 sec
 ARIMA(3,1,3)(0,0,0)[0]             : AIC=-7801.584, Time=3.49 sec
 ARIMA(2,1,3)(0,0,0)[0]             : AIC=-7803.638, Time=3.47 sec
 ARIMA(4,1,3)(0,0,0)[0]             : AIC=-7797.230, Time=3.16 sec

Best model:  ARIMA(3,1,2)(0,0,0)[0]
Total fit time: 73.559 seconds
Selected Order (p,d,q): (3, 1, 2)
                               SARIMAX Results

==========================================================================
========
Dep. Variable:            Temperature   No. Observations:
7304
Model:                 ARIMA(3, 1, 2)   Log Likelihood
3911.730
Date:                Tue, 08 Apr 2025   AIC                              -
7811.460
Time:                        16:15:44   BIC                              -
7770.084
Sample:                             0   HQIC                             -
7797.233
                                - 7304

Covariance Type:                  opg

==========================================================================
========
                 coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------
--------
ar.L1          1.6683      0.133     12.513      0.000       1.407
1.930
ar.L2         -0.9191      0.149     -6.167      0.000      -1.211
```

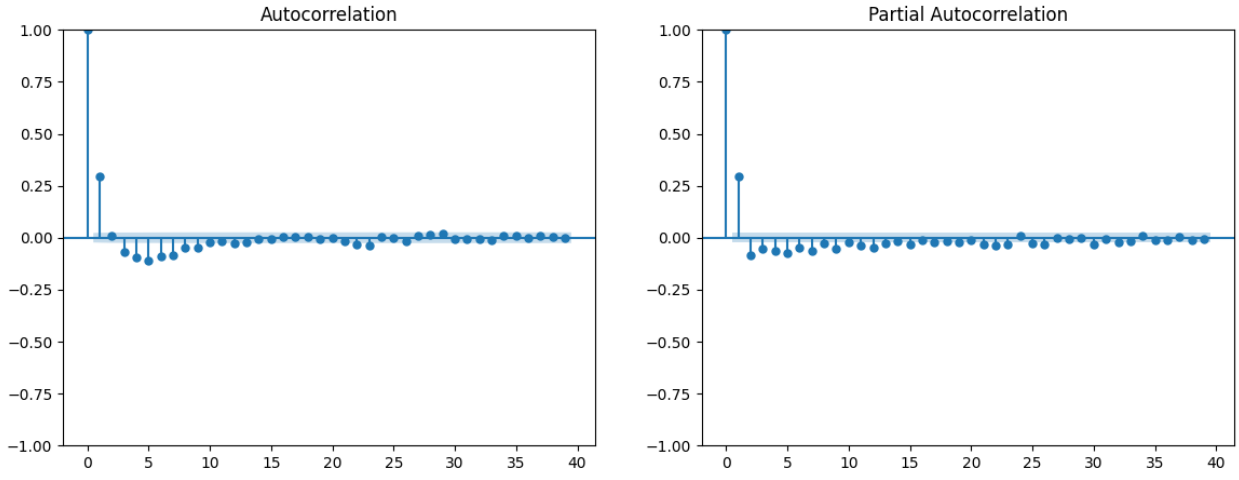
```
-0.627
ar.L3          0.1943      0.035      5.615      0.000        0.126
0.262
ma.L1         -1.4803      0.135    -10.984      0.000       -1.744
-1.216
ma.L2          0.4939      0.130      3.797      0.000        0.239
0.749
sigma2         0.0201      0.000     69.269      0.000        0.020
0.021
===================================================================
=============
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):
134.73
Prob(Q):                              0.91   Prob(JB):
0.00
Heteroskedasticity (H):               0.98   Skew:
-0.05
Prob(H) (two-sided):                  0.67   Kurtosis:
3.66
===================================================================
=============

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
```




```
ARIMA MAE (1900-1920): 0.3136, RMSE: 0.3242

Analyzing period: 1920-1940
```



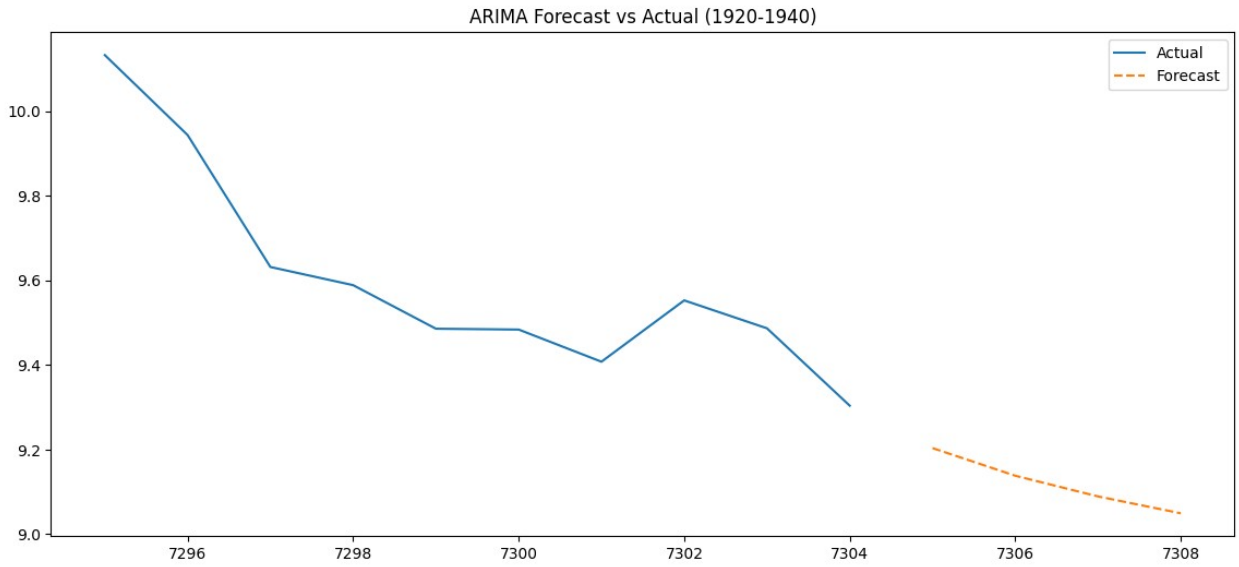

```
ADF Statistic: -13.478767214028714
p-value: 3.2724619072701088e-25
Stationary series (p <= 0.05)
ADF Statistic: -22.837800799250253
p-value: 0.0
Stationary series (p <= 0.05)
```



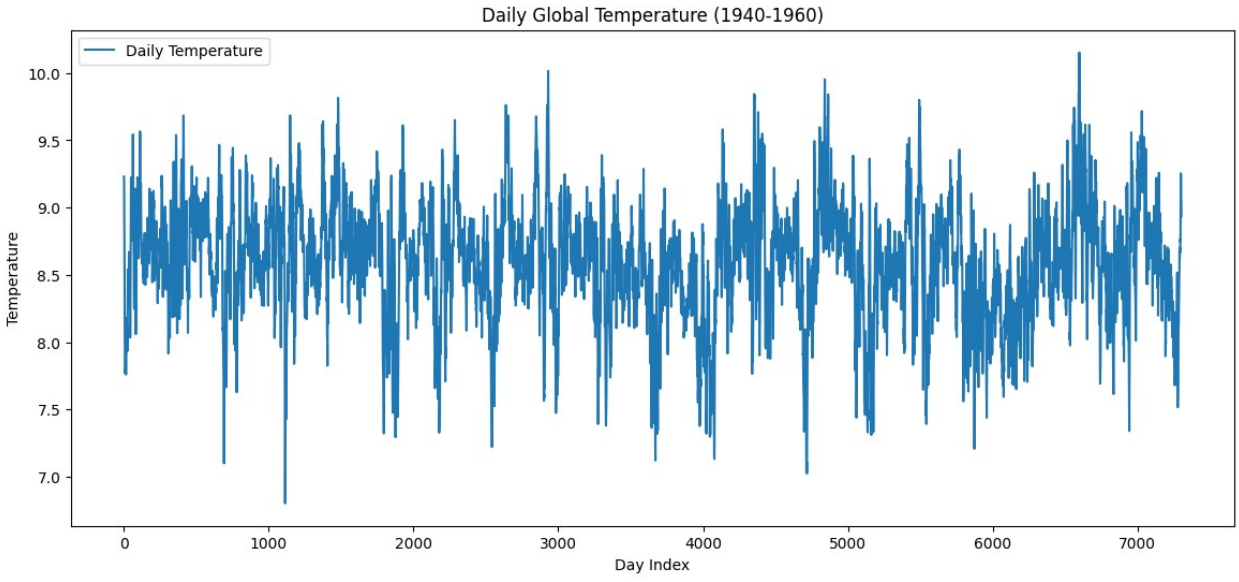

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=-9226.554, Time=4.12 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-8286.312, Time=0.30 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-8955.796, Time=1.40 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-8989.146, Time=0.54 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=-8288.309, Time=0.14 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=-8999.913, Time=3.03 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=-9268.339, Time=3.10 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=-8995.692, Time=0.78 sec
```

```
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=-9009.027, Time=0.29 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=4.15 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=-9025.861, Time=0.77 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=inf, Time=6.89 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=inf, Time=1.32 sec

Best model:  ARIMA(2,1,1)(0,0,0)[0] intercept
Total fit time: 26.883 seconds
Selected Order (p,d,q): (2, 1, 1)
                               SARIMAX Results

==============================================================================
========
Dep. Variable:            Temperature   No. Observations:
7305
Model:                 ARIMA(2, 1, 1)   Log Likelihood
4659.833
Date:                Tue, 08 Apr 2025   AIC                              -
9311.666
Time:                        16:16:14   BIC                              -
9284.081
Sample:                             0   HQIC                             -
9302.181
                                - 7305

Covariance Type:                  opg

==============================================================================
========
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
--------
ar.L1          1.2597      0.010    128.984      0.000       1.241
1.279
ar.L2         -0.3330      0.010    -33.819      0.000      -0.352
-0.314
ma.L1         -0.9957      0.001   -862.638      0.000      -0.998
-0.993
sigma2         0.0163      0.000     68.804      0.000       0.016
0.017
==============================================================================
=============
Ljung-Box (L1) (Q):                   1.09   Jarque-Bera (JB):
124.05
Prob(Q):                              0.30   Prob(JB):
0.00
Heteroskedasticity (H):               0.81   Skew:
-0.10
Prob(H) (two-sided):                  0.00   Kurtosis:
```

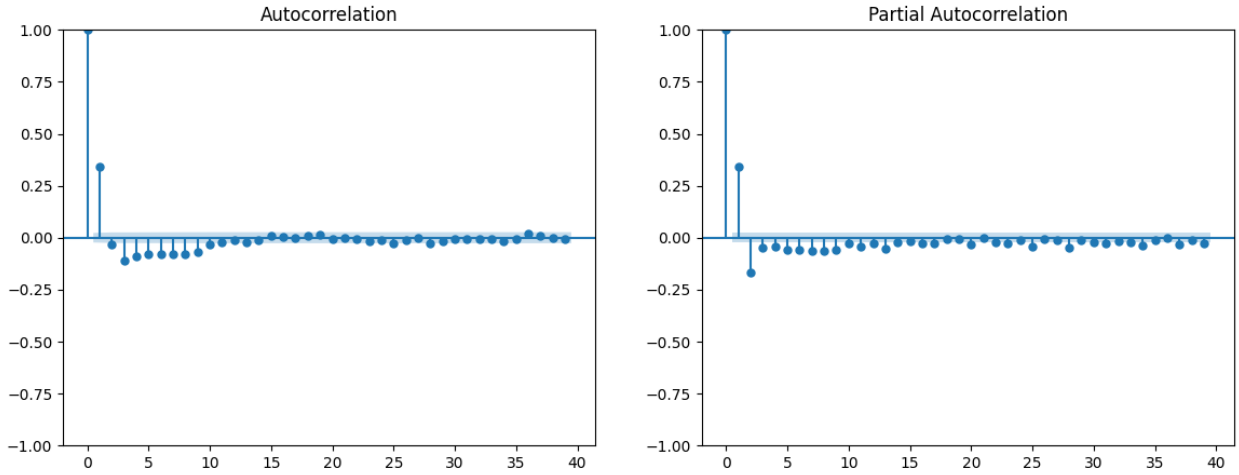
```
3.61
==============================================================================
==============

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
```




```
ARIMA MAE (1920-1940): 0.3177, RMSE: 0.3303

Analyzing period: 1940-1960
```

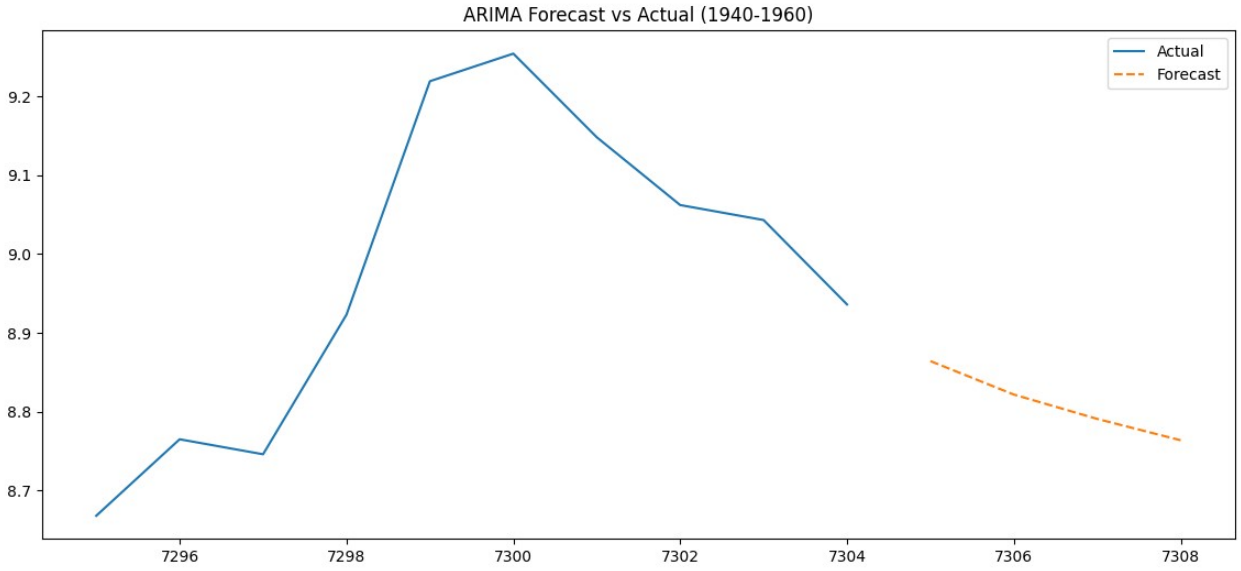


```
ADF Statistic: -12.27529371649404
p-value: 8.459582293576229e-23
Stationary series (p <= 0.05)
ADF Statistic: -20.324966838133054
p-value: 0.0
Stationary series (p <= 0.05)
```

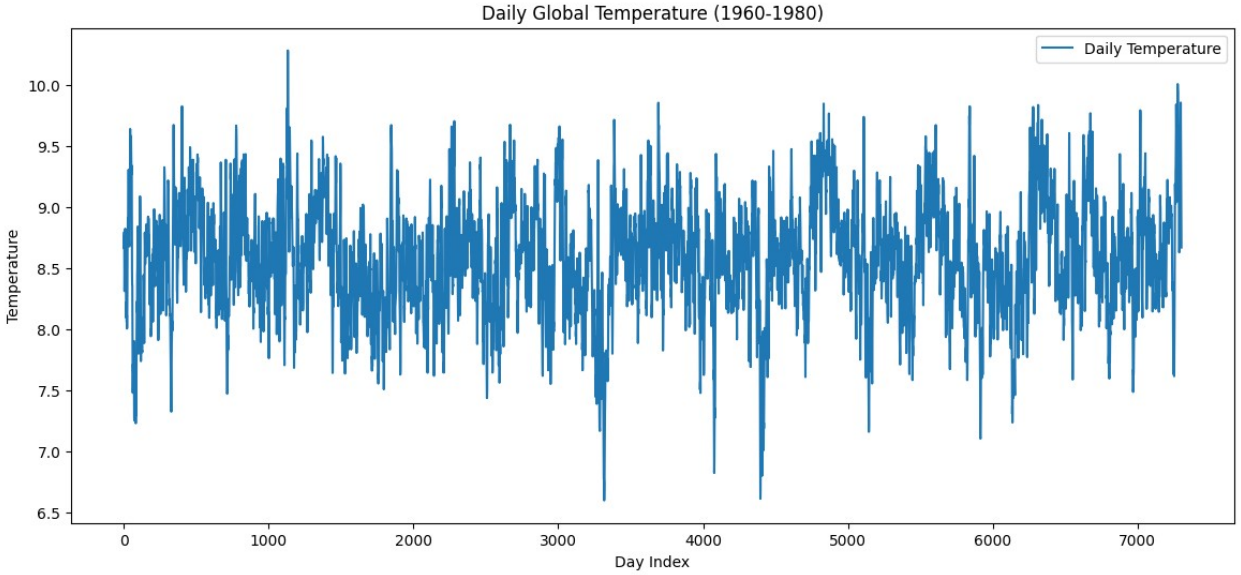



```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=-9197.336, Time=3.74 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-7804.979, Time=0.31 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-8696.580, Time=0.67 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-8841.272, Time=0.58 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=-7806.978, Time=0.22 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=-9110.173, Time=2.61 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=-9131.656, Time=2.43 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=-9165.646, Time=4.32 sec
 ARIMA(2,1,3)(0,0,0)[0] intercept   : AIC=inf, Time=5.45 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=-8843.897, Time=0.91 sec
 ARIMA(1,1,3)(0,0,0)[0] intercept   : AIC=inf, Time=5.18 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=-9201.021, Time=4.94 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=-8910.869, Time=0.38 sec
 ARIMA(4,1,1)(0,0,0)[0] intercept   : AIC=-9186.042, Time=4.95 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=-8896.629, Time=0.30 sec
 ARIMA(4,1,0)(0,0,0)[0] intercept   : AIC=-8922.922, Time=0.54 sec
 ARIMA(4,1,2)(0,0,0)[0] intercept   : AIC=-9106.094, Time=5.54 sec
 ARIMA(3,1,1)(0,0,0)[0]             : AIC=inf, Time=1.94 sec

Best model:  ARIMA(3,1,1)(0,0,0)[0] intercept
Total fit time: 45.012 seconds
Selected Order (p,d,q): (3, 1, 1)
                               SARIMAX Results

==============================================================================
========
```

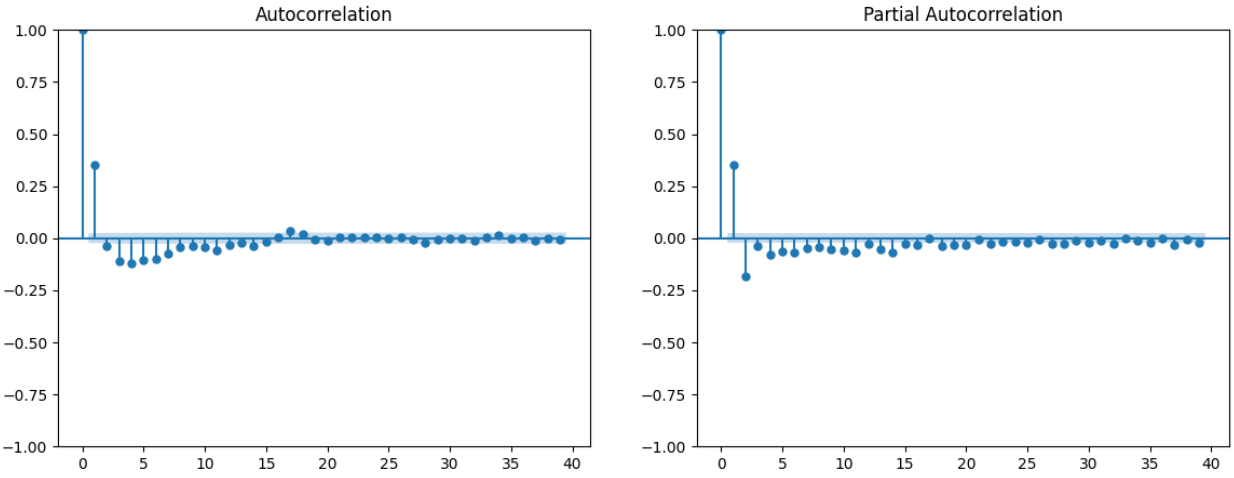
```
Dep. Variable:            Temperature   No. Observations:
7305
Model:                 ARIMA(3, 1, 1)   Log Likelihood
4610.040
Date:                Tue, 08 Apr 2025   AIC                              -
9210.079
Time:                        16:17:02   BIC                              -
9175.599
Sample:                             0   HQIC                             -
9198.223
                                - 7305

Covariance Type:                  opg

==============================================================================
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
ar.L1          1.3389      0.011    121.546      0.000       1.317
1.361
ar.L2         -0.5224      0.018    -29.676      0.000      -0.557
-0.488
ar.L3          0.1090      0.011      9.966      0.000       0.088
0.130
ma.L1         -0.9937      0.001   -685.111      0.000      -0.997
-0.991
sigma2         0.0166      0.000     65.655      0.000       0.016
0.017
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
45.52
Prob(Q):                              0.99   Prob(JB):
0.00
Heteroskedasticity (H):               1.14   Skew:
-0.06
Prob(H) (two-sided):                  0.00   Kurtosis:
3.36
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
```




```
ARIMA MAE (1940-1960): 0.2372, RMSE: 0.2406

Analyzing period: 1960-1980
```



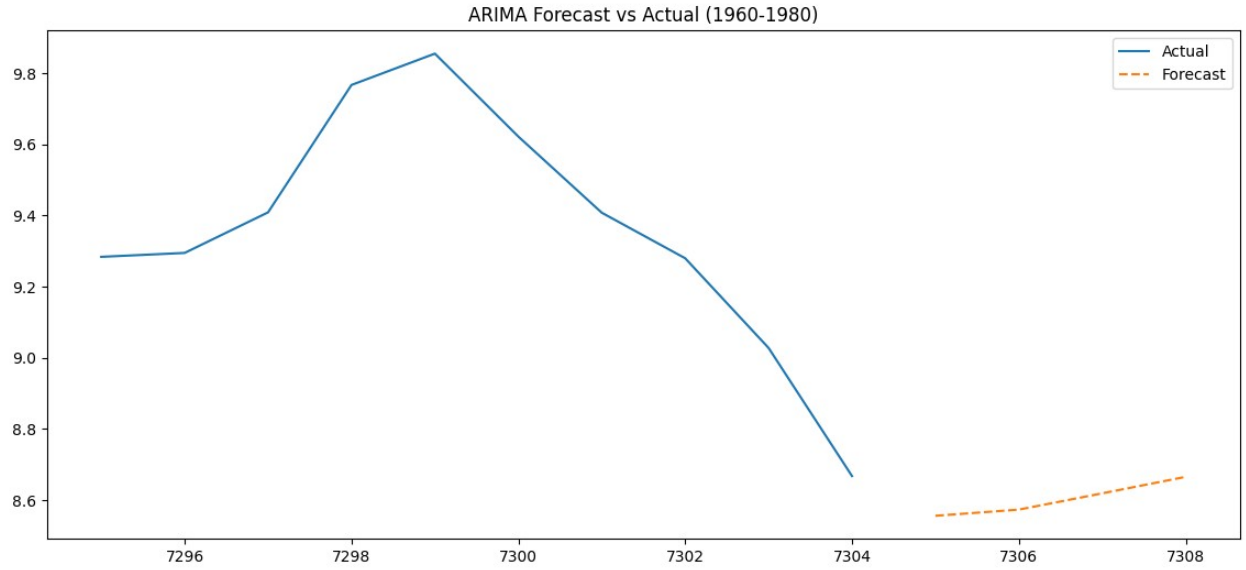

```
ADF Statistic: -11.993721933526885
p-value: 3.4729483329151113e-22
Stationary series (p <= 0.05)
ADF Statistic: -20.037106267414817
p-value: 0.0
Stationary series (p <= 0.05)
```



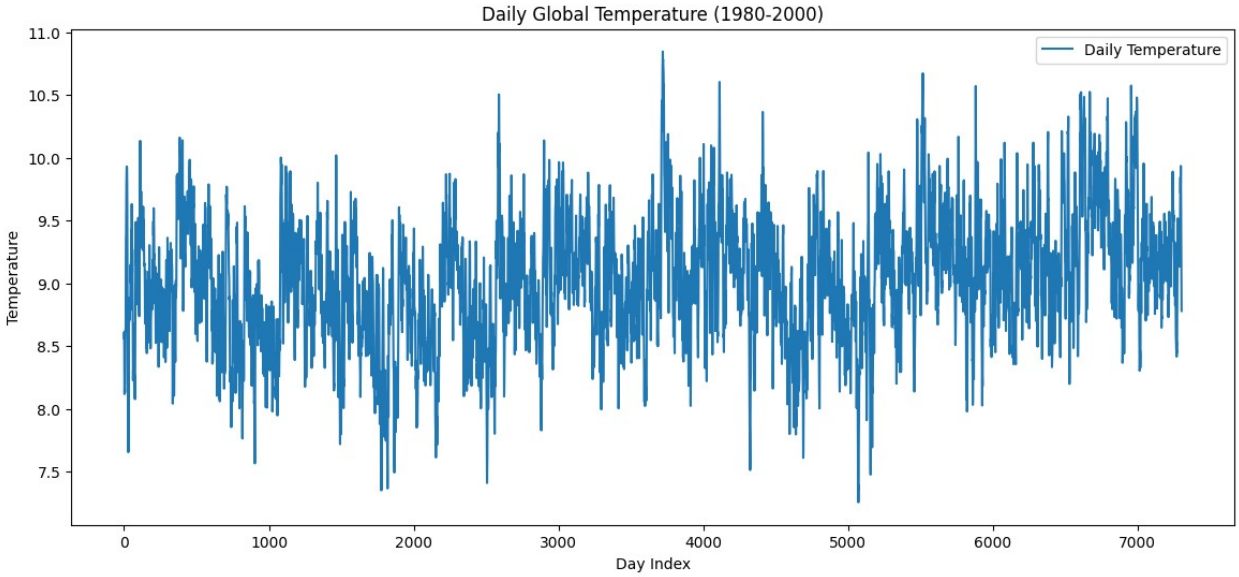

```
Performing stepwise search to minimize aic
 ARIMA(2,0,2)(0,0,0)[0]             : AIC=-7113.495, Time=2.74 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=52160.627, Time=0.04 sec
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=inf, Time=0.27 sec
 ARIMA(0,0,1)(0,0,0)[0]             : AIC=inf, Time=0.32 sec
 ARIMA(1,0,2)(0,0,0)[0]             : AIC=-7118.198, Time=0.73 sec
 ARIMA(0,0,2)(0,0,0)[0]             : AIC=32969.431, Time=0.49 sec
 ARIMA(1,0,1)(0,0,0)[0]             : AIC=-7117.974, Time=0.23 sec
 ARIMA(1,0,3)(0,0,0)[0]             : AIC=-7144.448, Time=0.96 sec
 ARIMA(0,0,3)(0,0,0)[0]             : AIC=25465.631, Time=0.91 sec
 ARIMA(2,0,3)(0,0,0)[0]             : AIC=-7114.261, Time=2.36 sec
 ARIMA(1,0,4)(0,0,0)[0]             : AIC=-7207.252, Time=0.78 sec
 ARIMA(0,0,4)(0,0,0)[0]             : AIC=19531.028, Time=1.38 sec
 ARIMA(2,0,4)(0,0,0)[0]             : AIC=-7140.346, Time=3.28 sec
 ARIMA(1,0,5)(0,0,0)[0]             : AIC=-7270.931, Time=1.07 sec
 ARIMA(0,0,5)(0,0,0)[0]             : AIC=15142.056, Time=2.12 sec
 ARIMA(2,0,5)(0,0,0)[0]             : AIC=-7203.939, Time=4.29 sec
 ARIMA(1,0,6)(0,0,0)[0]             : AIC=-7353.574, Time=1.52 sec
 ARIMA(0,0,6)(0,0,0)[0]             : AIC=11626.131, Time=2.00 sec
 ARIMA(2,0,6)(0,0,0)[0]             : AIC=-7536.162, Time=2.82 sec
 ARIMA(3,0,6)(0,0,0)[0]             : AIC=-7212.935, Time=5.59 sec
 ARIMA(3,0,5)(0,0,0)[0]             : AIC=-7155.627, Time=4.42 sec
 ARIMA(2,0,6)(0,0,0)[0] intercept   : AIC=-7550.224, Time=6.41 sec
 ARIMA(1,0,6)(0,0,0)[0] intercept   : AIC=-7542.015, Time=9.12 sec
 ARIMA(2,0,5)(0,0,0)[0] intercept   : AIC=-7542.308, Time=6.02 sec
 ARIMA(3,0,6)(0,0,0)[0] intercept   : AIC=-7555.912, Time=9.39 sec
 ARIMA(3,0,5)(0,0,0)[0] intercept   : AIC=-7554.213, Time=9.39 sec
 ARIMA(4,0,6)(0,0,0)[0] intercept   : AIC=-7556.616, Time=11.34 sec
 ARIMA(4,0,5)(0,0,0)[0] intercept   : AIC=-7556.238, Time=8.57 sec
 ARIMA(5,0,6)(0,0,0)[0] intercept   : AIC=-7556.134, Time=11.71 sec
 ARIMA(5,0,5)(0,0,0)[0] intercept   : AIC=-7543.517, Time=11.42 sec
 ARIMA(4,0,6)(0,0,0)[0]             : AIC=-7140.561, Time=5.66 sec

Best model:  ARIMA(4,0,6)(0,0,0)[0] intercept
```

```
Total fit time: 127.404 seconds
Selected Order (p,d,q): (4, 0, 6)
                               SARIMAX Results

==========================================================================
========
Dep. Variable:            Temperature   No. Observations:
7305
Model:                 ARIMA(4, 0, 6)   Log Likelihood
3785.042
Date:                Tue, 08 Apr 2025   AIC                            -
7546.084
Time:                        16:19:23   BIC                            -
7463.328
Sample:                             0   HQIC                           -
7517.629
                                - 7305

Covariance Type:                  opg

==========================================================================
========
                 coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------
--------
const          8.5912      0.027    319.794      0.000       8.539
8.644
ar.L1          0.4649      0.160      2.909      0.004       0.152
0.778
ar.L2          0.2813      0.204      1.378      0.168      -0.119
0.681
ar.L3          0.6666      0.206      3.233      0.001       0.262
1.071
ar.L4         -0.4816      0.123     -3.909      0.000      -0.723
-0.240
ma.L1          0.8999      0.160      5.629      0.000       0.587
1.213
ma.L2          0.3656      0.166      2.203      0.028       0.040
0.691
ma.L3         -0.5479      0.141     -3.883      0.000      -0.825
-0.271
ma.L4         -0.3966      0.069     -5.774      0.000      -0.531
-0.262
ma.L5         -0.1670      0.027     -6.194      0.000      -0.220
-0.114
ma.L6         -0.0677      0.015     -4.508      0.000      -0.097
-0.038
sigma2         0.0209      0.000     62.788      0.000       0.020
0.022
```

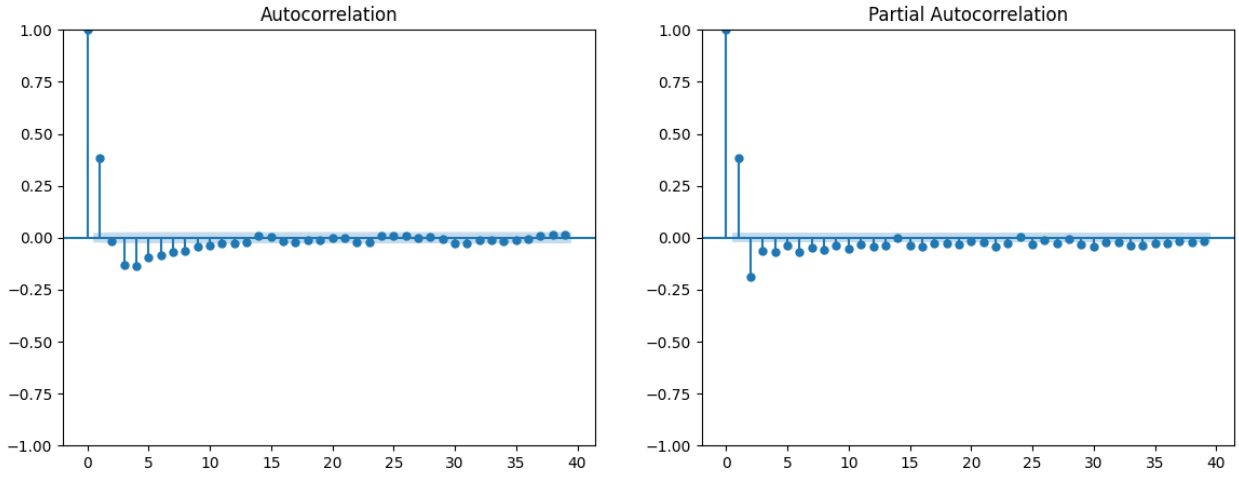
```
================================================================================
==============
Ljung-Box (L1) (Q):                   0.64   Jarque-Bera (JB):
16.64
Prob(Q):                              0.42   Prob(JB):
0.00
Heteroskedasticity (H):               1.05   Skew:
-0.04
Prob(H) (two-sided):                  0.20   Kurtosis:
3.22
================================================================================
==============

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
```




```
ARIMA MAE (1960-1980): 0.4921, RMSE: 0.5897

Analyzing period: 1980-2000
```



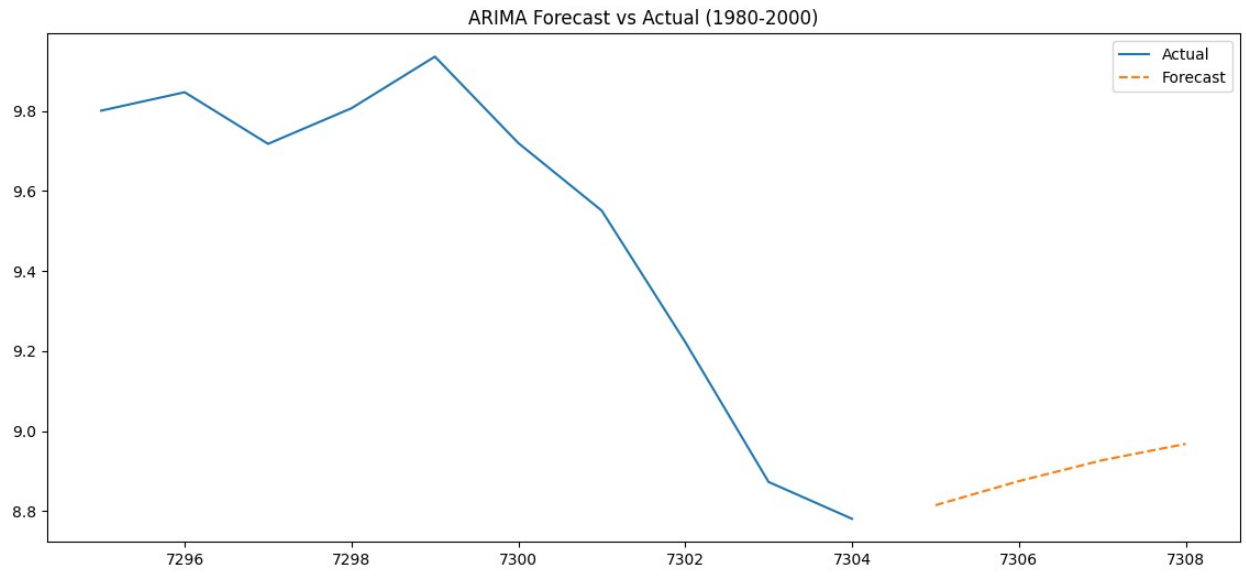

```
ADF Statistic: -7.820413456227168
p-value: 6.693545479622451e-12
Stationary series (p <= 0.05)
ADF Statistic: -21.99891719650499
p-value: 0.0
Stationary series (p <= 0.05)
```



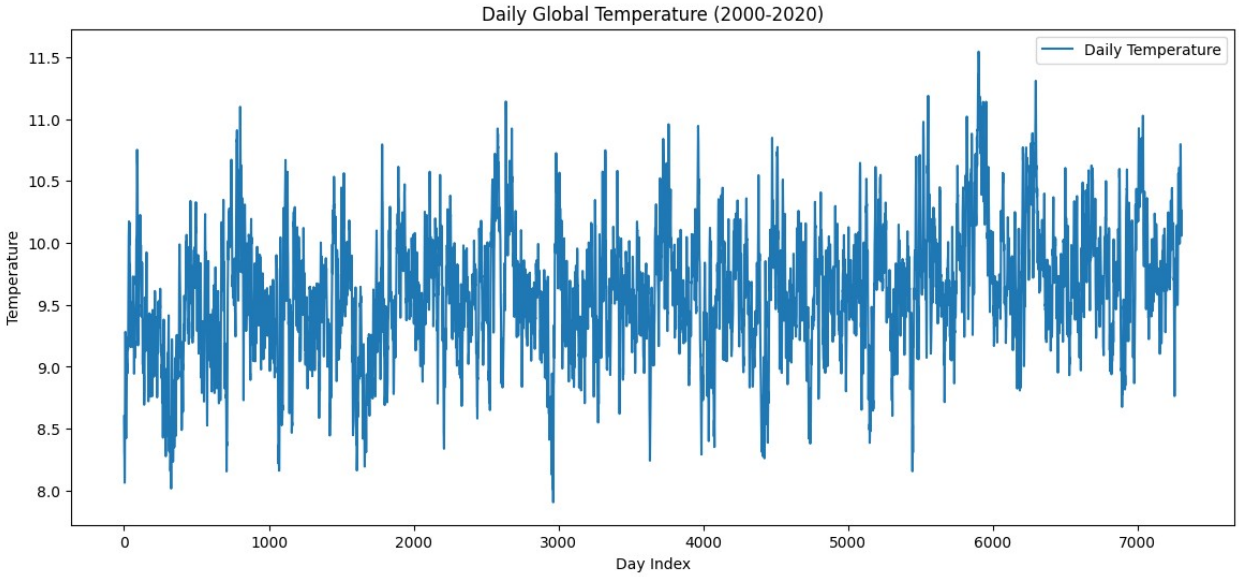

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=-7568.907, Time=3.75 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-5799.393, Time=0.41 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-6941.991, Time=0.76 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-7123.404, Time=0.53 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=-5801.392, Time=0.15 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=inf, Time=3.84 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=-7483.240, Time=3.38 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=-7505.095, Time=4.59 sec
```

```
 ARIMA(2,1,3)(0,0,0)[0] intercept   : AIC=-7550.989, Time=5.04 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=-7136.362, Time=1.01 sec
 ARIMA(1,1,3)(0,0,0)[0] intercept   : AIC=inf, Time=3.21 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=-7572.700, Time=4.65 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=-7229.854, Time=0.45 sec
 ARIMA(4,1,1)(0,0,0)[0] intercept   : AIC=-7551.042, Time=4.83 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=-7204.426, Time=0.33 sec
 ARIMA(4,1,0)(0,0,0)[0] intercept   : AIC=-7264.488, Time=0.55 sec
 ARIMA(4,1,2)(0,0,0)[0] intercept   : AIC=-7562.450, Time=6.00 sec
 ARIMA(3,1,1)(0,0,0)[0]             : AIC=-7575.347, Time=1.84 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=-7485.235, Time=1.17 sec
 ARIMA(3,1,0)(0,0,0)[0]             : AIC=-7231.853, Time=0.71 sec
 ARIMA(4,1,1)(0,0,0)[0]             : AIC=-7573.350, Time=2.56 sec
 ARIMA(3,1,2)(0,0,0)[0]             : AIC=-7573.293, Time=2.92 sec
 ARIMA(2,1,0)(0,0,0)[0]             : AIC=-7206.426, Time=0.81 sec
 ARIMA(2,1,2)(0,0,0)[0]             : AIC=-7571.295, Time=1.82 sec
 ARIMA(4,1,0)(0,0,0)[0]             : AIC=-7266.487, Time=0.27 sec
 ARIMA(4,1,2)(0,0,0)[0]             : AIC=-7573.069, Time=3.14 sec

Best model:  ARIMA(3,1,1)(0,0,0)[0]
Total fit time: 58.757 seconds
Selected Order (p,d,q): (3, 1, 1)
                               SARIMAX Results

==============================================================================
========
Dep. Variable:            Temperature   No. Observations:
7305
Model:                 ARIMA(3, 1, 1)   Log Likelihood
3792.673
Date:                Tue, 08 Apr 2025   AIC                              -
7575.347
Time:                        16:20:25   BIC                              -
7540.866
Sample:                             0   HQIC                             -
7563.491
                                - 7305

Covariance Type:                  opg

==============================================================================
========
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
--------
ar.L1          1.3794      0.012    119.194      0.000       1.357
1.402
ar.L2         -0.5858      0.018    -31.970      0.000      -0.622
-0.550
```

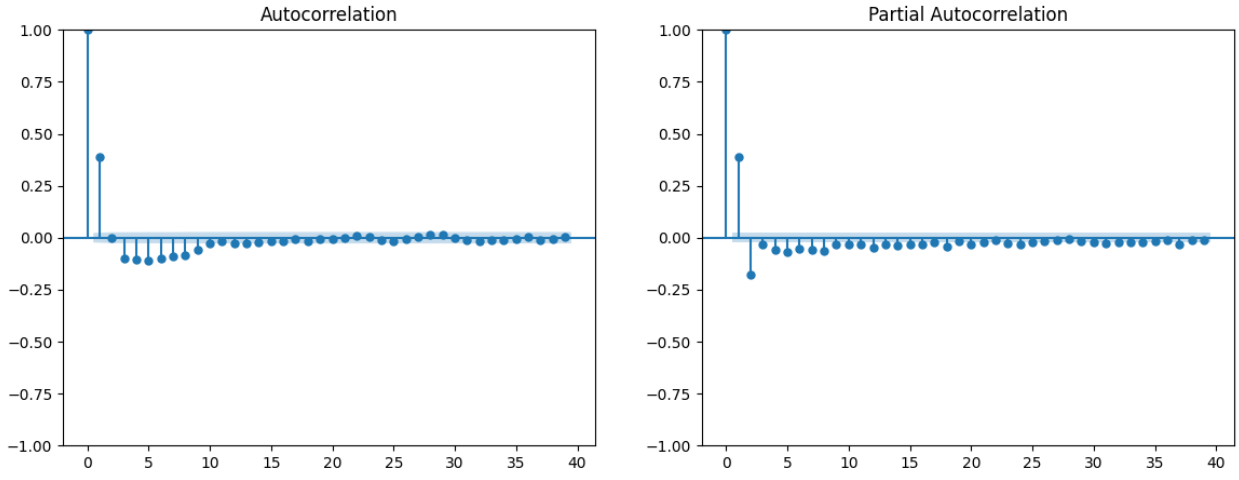
```
ar.L3          0.1164      0.012      9.993      0.000       0.094
0.139
ma.L1         -0.9875      0.002   -443.015      0.000      -0.992
-0.983
sigma2         0.0207      0.000     62.051      0.000       0.020
0.021
===================================================================
=============
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
5.47
Prob(Q):                              1.00   Prob(JB):
0.07
Heteroskedasticity (H):               1.09   Skew:
0.03
Prob(H) (two-sided):                  0.03   Kurtosis:
3.11
===================================================================
=============

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
```




```
ARIMA MAE (1980-2000): 0.3313, RMSE: 0.4185

Analyzing period: 2000-2020
```



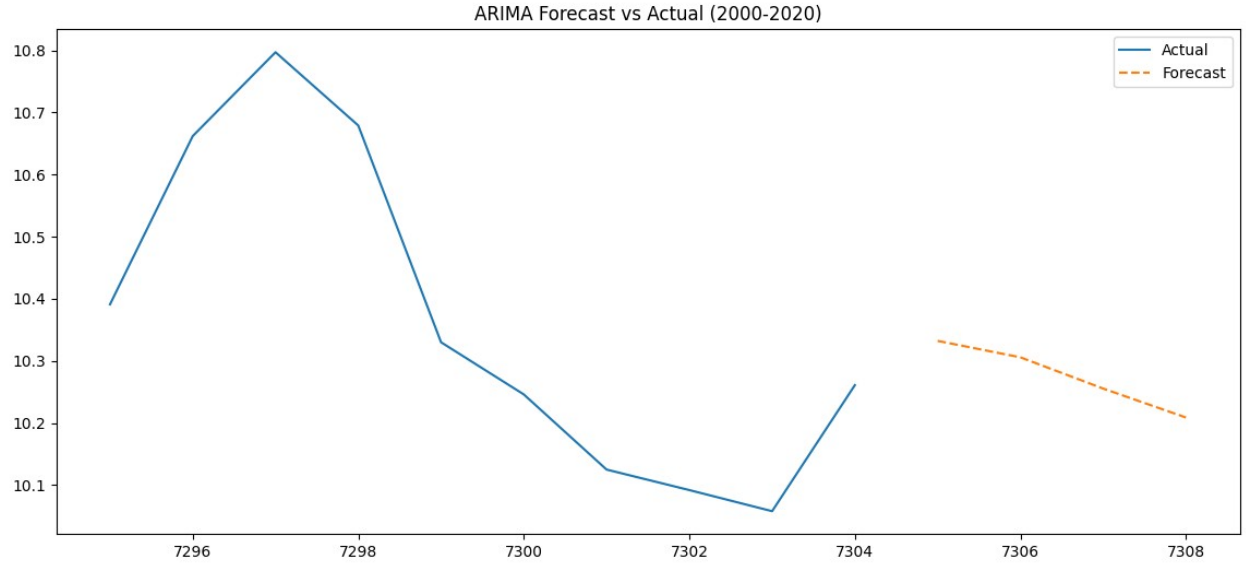

```
ADF Statistic: -14.245515559630434
p-value: 1.5119106986016173e-26
Stationary series (p <= 0.05)
ADF Statistic: -20.557143402726886
p-value: 0.0
Stationary series (p <= 0.05)
```




```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=inf, Time=6.59 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-6194.864, Time=0.32 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-7387.833, Time=0.18 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-7558.217, Time=0.52 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=-6196.849, Time=0.19 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=-7577.079, Time=0.91 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=-7864.979, Time=3.34 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=-7624.647, Time=0.75 sec
```

```
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=4.58 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=inf, Time=3.46 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=-7630.980, Time=0.45 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=-7880.396, Time=4.77 sec
 ARIMA(4,1,2)(0,0,0)[0] intercept   : AIC=-7927.260, Time=6.13 sec
 ARIMA(4,1,1)(0,0,0)[0] intercept   : AIC=-7879.431, Time=6.29 sec
 ARIMA(5,1,2)(0,0,0)[0] intercept   : AIC=-7924.450, Time=7.32 sec
 ARIMA(4,1,3)(0,0,0)[0] intercept   : AIC=-7948.695, Time=6.21 sec
 ARIMA(3,1,3)(0,0,0)[0] intercept   : AIC=-7948.723, Time=5.66 sec
 ARIMA(2,1,3)(0,0,0)[0] intercept   : AIC=inf, Time=5.12 sec
 ARIMA(3,1,4)(0,0,0)[0] intercept   : AIC=inf, Time=5.02 sec
 ARIMA(2,1,4)(0,0,0)[0] intercept   : AIC=-7951.121, Time=7.82 sec
 ARIMA(1,1,4)(0,0,0)[0] intercept   : AIC=-7952.044, Time=5.30 sec
 ARIMA(0,1,4)(0,0,0)[0] intercept   : AIC=-7628.879, Time=1.97 sec
 ARIMA(1,1,3)(0,0,0)[0] intercept   : AIC=inf, Time=6.44 sec
 ARIMA(1,1,5)(0,0,0)[0] intercept   : AIC=-7929.124, Time=6.24 sec
 ARIMA(0,1,3)(0,0,0)[0] intercept   : AIC=-7602.985, Time=1.69 sec
 ARIMA(0,1,5)(0,0,0)[0] intercept   : AIC=-7680.740, Time=2.74 sec
 ARIMA(2,1,5)(0,0,0)[0] intercept   : AIC=-7934.863, Time=7.41 sec
 ARIMA(1,1,4)(0,0,0)[0]             : AIC=inf, Time=2.97 sec

Best model:  ARIMA(1,1,4)(0,0,0)[0] intercept
Total fit time: 110.410 seconds
Selected Order (p,d,q): (1, 1, 4)
                               SARIMAX Results

==============================================================================
========
Dep. Variable:            Temperature   No. Observations:
7305
Model:                 ARIMA(1, 1, 4)   Log Likelihood
3987.527
Date:                Tue, 08 Apr 2025   AIC                              -
7963.053
Time:                        16:22:19   BIC                              -
7921.676
Sample:                             0   HQIC                             -
7948.826
                                - 7305

Covariance Type:                  opg

==============================================================================
========
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
--------
ar.L1          0.8811      0.007    118.404      0.000       0.867
0.896
```

```
ma.L1          -0.4752      0.013    -35.700      0.000      -0.501
-0.449
ma.L2          -0.3923      0.013    -30.102      0.000      -0.418
-0.367
ma.L3          -0.1173      0.013     -8.985      0.000      -0.143
-0.092
ma.L4          -0.0098      0.013     -0.754      0.451      -0.035
0.016
sigma2          0.0196      0.000     63.814      0.000       0.019
0.020
===================================================================
=============
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
24.88
Prob(Q):                              0.99   Prob(JB):
0.00
Heteroskedasticity (H):               0.96   Skew:
0.07
Prob(H) (two-sided):                  0.29   Kurtosis:
3.25
===================================================================
=============

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
```




```
ARIMA MAE (2000-2020): 0.1676, RMSE: 0.1805

ARIMA Performance Summary:
  Time Period      Order       MAE      RMSE
```

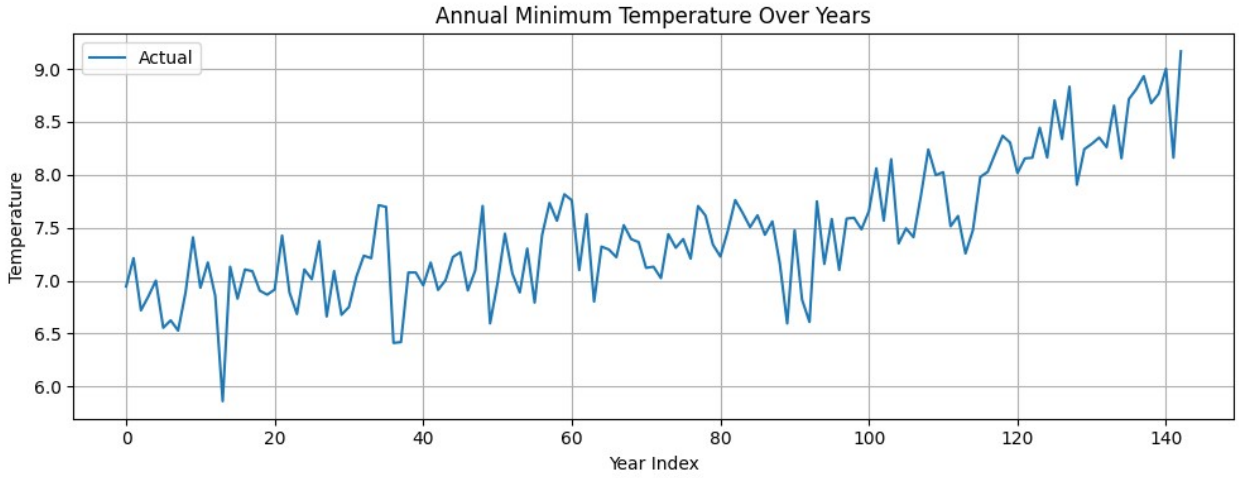
```
0   1880-1900  (1, 1, 3)  0.308529  0.326355
1   1900-1920  (3, 1, 2)  0.313560  0.324202
2   1920-1940  (2, 1, 1)  0.317736  0.330327
3   1940-1960  (3, 1, 1)  0.237159  0.240629
4   1960-1980  (4, 0, 6)  0.492115  0.589666
5   1980-2000  (3, 1, 1)  0.331345  0.418542
6   2000-2020  (1, 1, 4)  0.167644  0.180514
```

```
# todays work
#=== Import Libraries ===
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from pmdarima import auto_arima
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_absolute_error, mean_squared_error

# === Step 1: Prepare Data ===
# Assuming df already has 'Year', 'Month', 'Day', 'Temperature', and
'Anomaly' columns

# Convert to datetime and set as index
df['Date'] = pd.to_datetime(df[['Year', 'Month', 'Day']])
df.set_index('Date', inplace=True)

# Retain only needed columns
df = df[['Temperature', 'Anomaly']]

# Add 'Year' column for aggregation
df['Year'] = df.index.year

# === Step 2: Aggregate to Annual Min, Max, Mean ===
annual_stats = df.groupby('Year')['Temperature'].agg(['min', 'max',
'mean']).reset_index()
annual_stats.columns = ['Year', 'MinTemp', 'MaxTemp', 'MeanTemp']
print("Annual Temperature Summary:")
print(annual_stats.head())

# === Step 3: Define ADF Test Function ===
def adf_test(series, title=""):
    result = adfuller(series.dropna())
    print(f"\nADF Test - {title}")
    print(f"ADF Statistic: {result[0]:.4f}")
    print(f"p-value: {result[1]:.4f}")
    if result[1] > 0.05:
        print("=> Non-stationary (p > 0.05)")
    else:
        print("=> Stationary (p <= 0.05)")
```

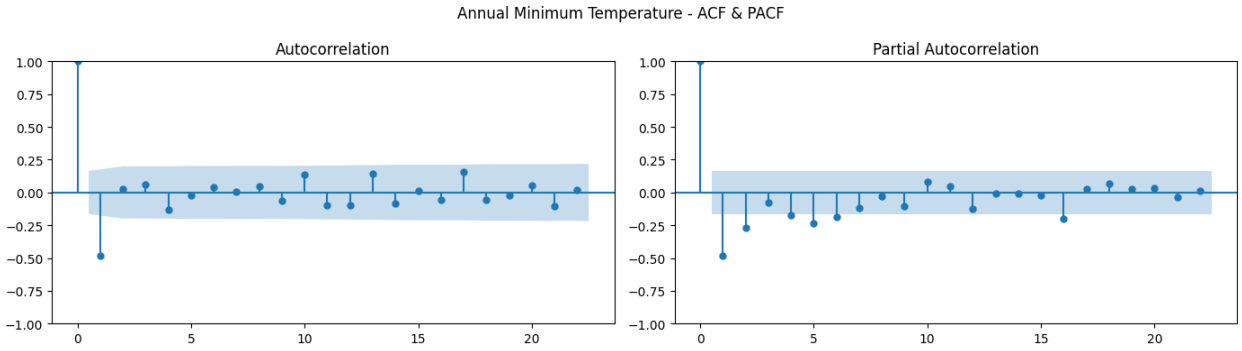
```
# === Step 4: Define Forecasting Function ===
def model_forecast(series, title, forecast_years=5):
    print(f"\n=== {title} ===")

    # Plot time series
    plt.figure(figsize=(10, 4))
    plt.plot(series.values, label='Actual')
    plt.title(f'{title} Over Years')
    plt.xlabel('Year Index')
    plt.ylabel('Temperature')
    plt.grid(True)
    plt.legend()
    plt.tight_layout()
    plt.show()

    # ADF Test
    adf_test(series, title)

    # Differencing
    series_diff = series.diff().dropna()

    # ACF and PACF
    fig, axes = plt.subplots(1, 2, figsize=(14, 4))
    plot_acf(series_diff, ax=axes[0])
    plot_pacf(series_diff, ax=axes[1])
    fig.suptitle(f'{title} - ACF & PACF')
    plt.tight_layout()
    plt.show()

    # Fit Auto ARIMA
    auto_model = auto_arima(series, seasonal=False, trace=True,
                            error_action='ignore',
suppress_warnings=True)
    print(f"Selected ARIMA Order: {auto_model.order}")

    # Fit ARIMA model
    model = ARIMA(series, order=auto_model.order)
    model_fit = model.fit()
    print(model_fit.summary())

    # Forecast
    forecast = model_fit.forecast(steps=forecast_years)
    forecast_index = np.arange(len(series), len(series) +
forecast_years)

    # Plot forecast
    plt.figure(figsize=(10, 4))
    plt.plot(series.values, label='Actual')
```

```
    plt.plot(forecast_index, forecast, label='Forecast',
linestyle='--', color='red')
    plt.title(f'{title} Forecast for {forecast_years} Years')
    plt.xlabel('Year Index')
    plt.ylabel('Temperature')
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()

    return forecast

# === Step 5: Apply to All Three Series ===
min_forecast = model_forecast(annual_stats['MinTemp'], "Annual Minimum
Temperature")
max_forecast = model_forecast(annual_stats['MaxTemp'], "Annual Maximum
Temperature")
mean_forecast = model_forecast(annual_stats['MeanTemp'], "Annual Mean
Temperature")

# === Step 6: Combine Forecasts into One Table ===
forecast_years = np.arange(annual_stats['Year'].iloc[-1] + 1,
annual_stats['Year'].iloc[-1] + 6)
forecast_df = pd.DataFrame({
    'Year': forecast_years,
    'Min Forecast': min_forecast,
    'Max Forecast': max_forecast,
    'Mean Forecast': mean_forecast
})

print("\nForecasted Annual Temperatures:")
print(forecast_df)
```

```
Annual Temperature Summary:
   Year  MinTemp  MaxTemp  MeanTemp
0  1880    6.945    8.666  7.995557
1  1881    7.213    9.006  8.255378
2  1882    6.720    9.325  8.116529
3  1883    6.852    8.735  7.995145
4  1884    7.001    8.786  7.801880

=== Annual Minimum Temperature ===
```

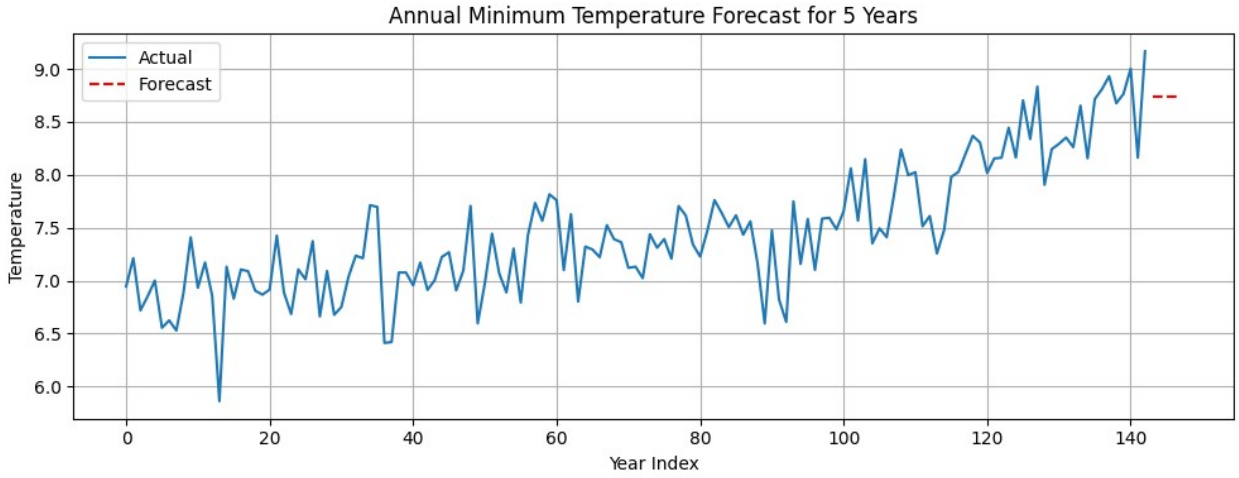



```
ADF Test - Annual Minimum Temperature
ADF Statistic: 0.1330
p-value: 0.9682
=> Non-stationary (p > 0.05)
```

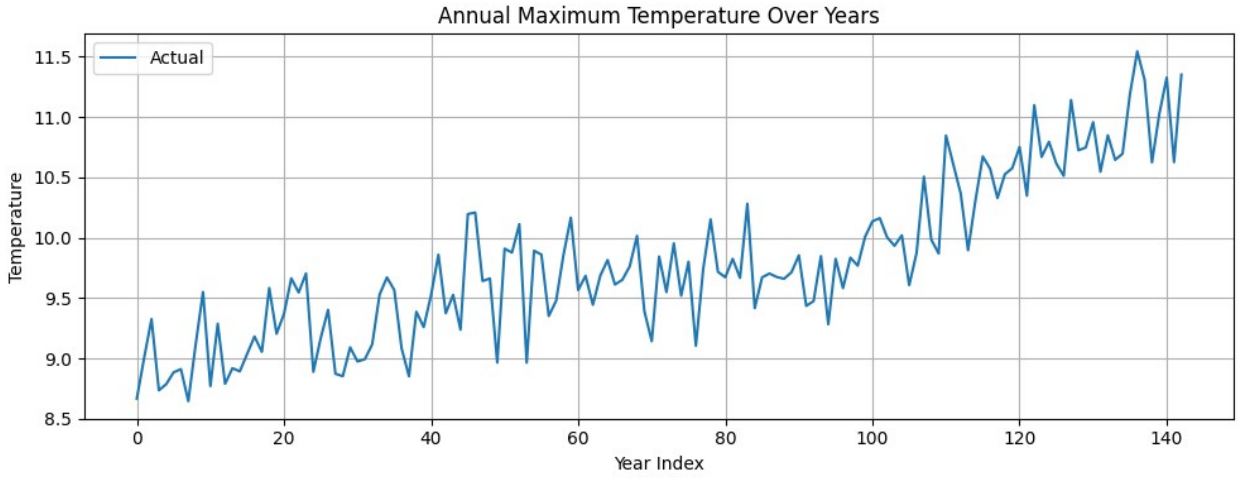



```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=104.664, Time=0.34 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=163.981, Time=0.03 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=126.994, Time=0.01 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=100.377, Time=0.06 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=162.173, Time=0.01 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=101.663, Time=0.08 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=101.781, Time=0.09 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=104.128, Time=0.09 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=103.725, Time=0.03 sec

Best model:  ARIMA(0,1,1)(0,0,0)[0] intercept
Total fit time: 0.754 seconds
Selected ARIMA Order: (0, 1, 1)
                               SARIMAX Results

==============================================================================
```

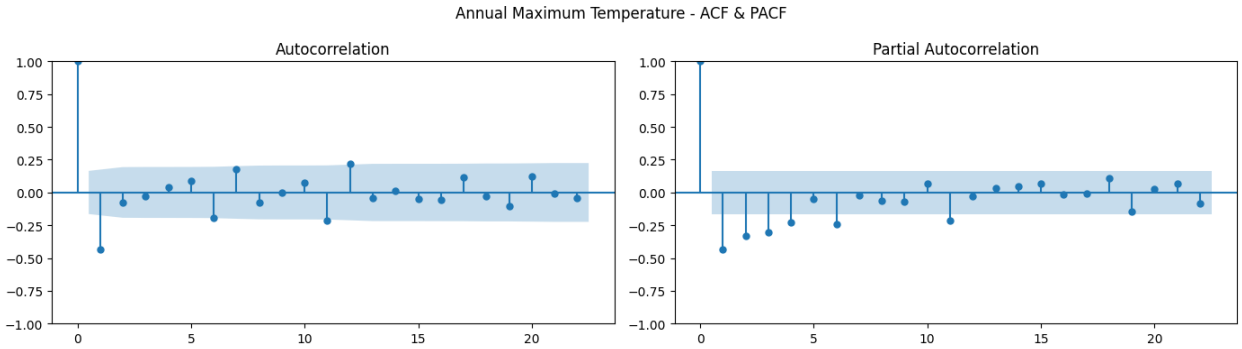
```
========
Dep. Variable:                 MinTemp   No. Observations:
143
Model:                  ARIMA(0, 1, 1)   Log Likelihood
-49.863
Date:                Tue, 06 May 2025   AIC
103.725
Time:                        14:34:53   BIC
109.637
Sample:                             0   HQIC
106.127
                                - 143

Covariance Type:                  opg

==============================================================================
========
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
--------
ma.L1         -0.7660      0.048    -16.102      0.000      -0.859
-0.673
sigma2         0.1174      0.014      8.355      0.000       0.090
0.145
==============================================================================
=============
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
8.93
Prob(Q):                              0.98   Prob(JB):
0.01
Heteroskedasticity (H):               0.94   Skew:
-0.57
Prob(H) (two-sided):                  0.82   Kurtosis:
3.44
==============================================================================
=============

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
```




```
=== Annual Maximum Temperature ===
```

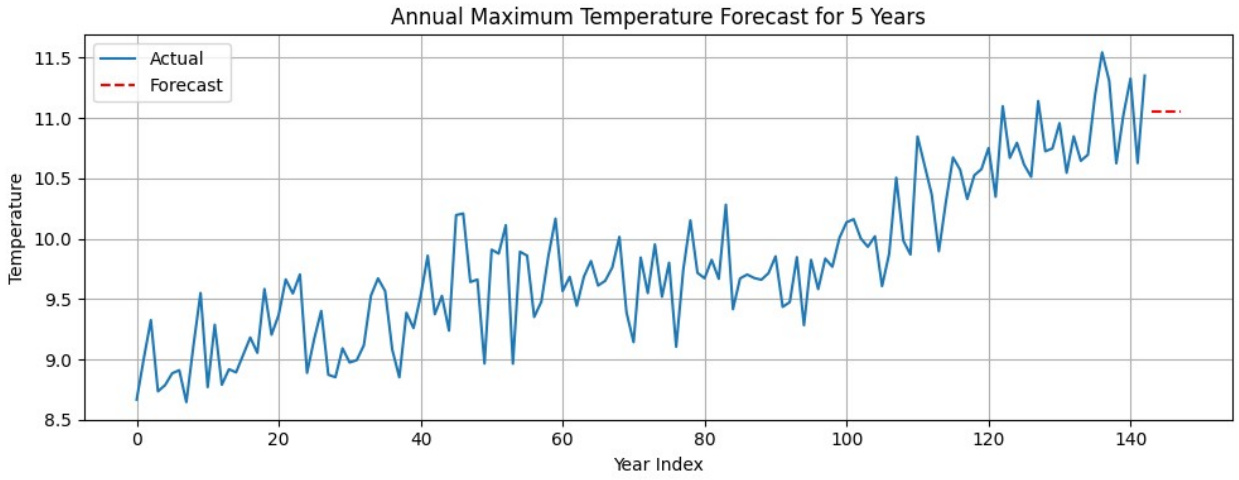



```
ADF Test - Annual Maximum Temperature
ADF Statistic: 0.8198
p-value: 0.9919
=> Non-stationary (p > 0.05)
```

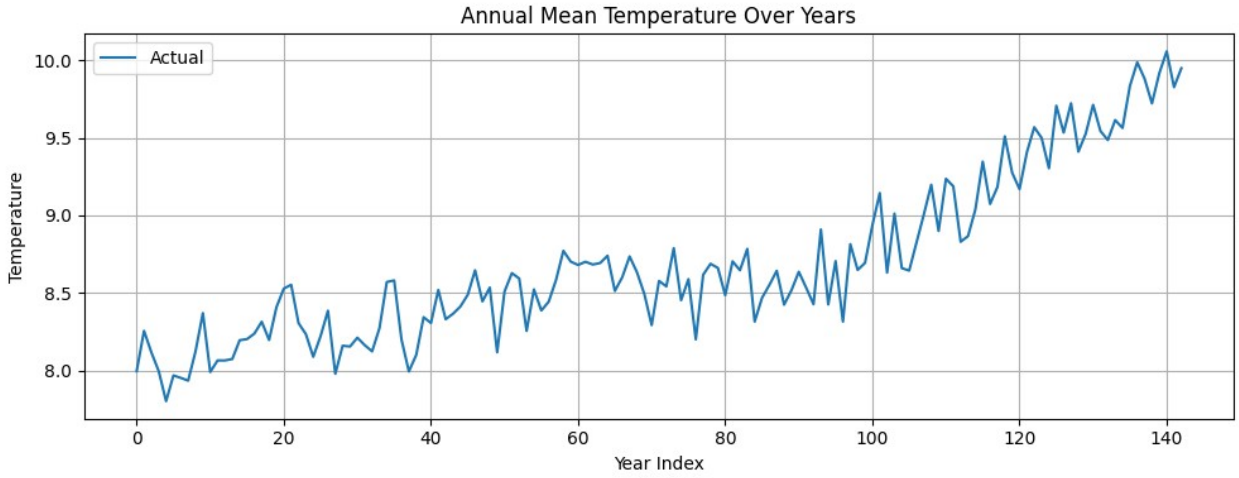



```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=82.085, Time=0.34 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=149.246, Time=0.02 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=120.757, Time=0.02 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=77.889, Time=0.06 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=147.557, Time=0.01 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=79.858, Time=0.11 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=79.849, Time=0.21 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=inf, Time=0.25 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=84.407, Time=0.03 sec

Best model:  ARIMA(0,1,1)(0,0,0)[0] intercept
Total fit time: 1.068 seconds
Selected ARIMA Order: (0, 1, 1)
                               SARIMAX Results

==============================================================================
========
Dep. Variable:                 MaxTemp   No. Observations:
143
Model:                 ARIMA(0, 1, 1)   Log Likelihood
-40.204
Date:                Tue, 06 May 2025   AIC
84.407
Time:                        14:34:55   BIC
90.319
Sample:                             0   HQIC
86.810
                                - 143

Covariance Type:                  opg

==============================================================================
========
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
--------
ma.L1         -0.7663      0.055    -14.022      0.000      -0.873
```

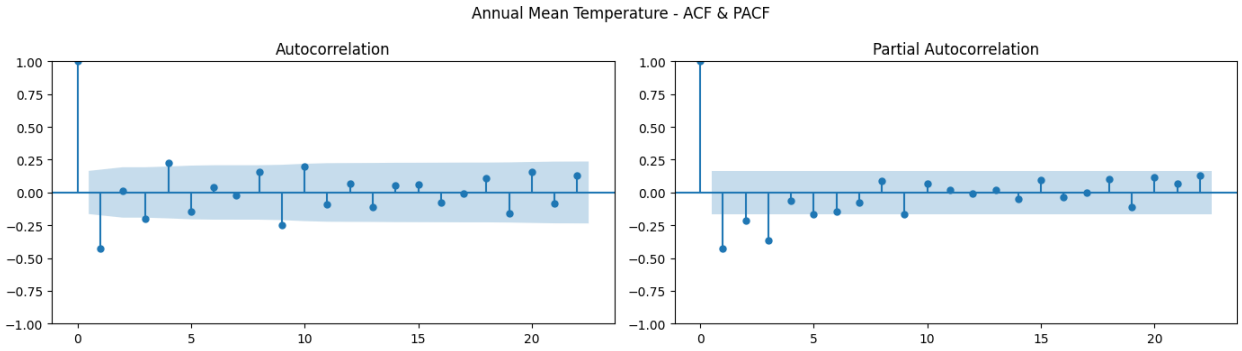
```
-0.659
sigma2         0.1025      0.012      8.216      0.000       0.078
0.127
===================================================================================
=============
Ljung-Box (L1) (Q):                   0.33   Jarque-Bera (JB):
0.11
Prob(Q):                              0.57   Prob(JB):
0.94
Heteroskedasticity (H):               0.97   Skew:
-0.07
Prob(H) (two-sided):                  0.92   Kurtosis:
2.95
===================================================================================
=============

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
```




```
=== Annual Mean Temperature ===
```

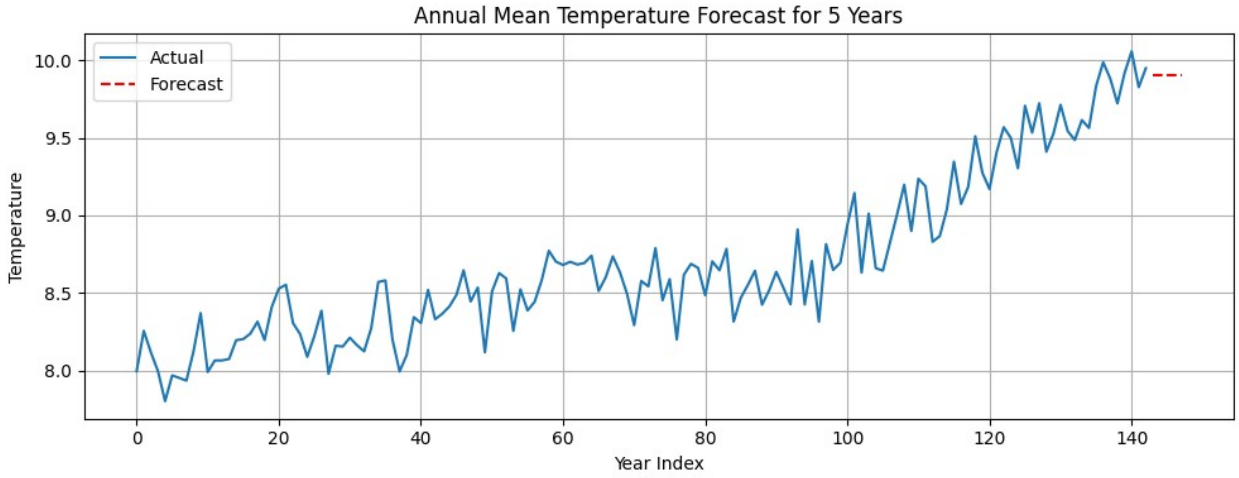



```
ADF Test - Annual Mean Temperature
ADF Statistic: 1.7800
p-value: 0.9983
=> Non-stationary (p > 0.05)
```




```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=-83.595, Time=0.22 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-32.737, Time=0.04 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-59.463, Time=0.04 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-84.205, Time=0.06 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=-34.143, Time=0.01 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=-82.957, Time=0.09 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=-82.956, Time=0.09 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=-82.726, Time=0.17 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=-78.140, Time=0.02 sec

Best model:  ARIMA(0,1,1)(0,0,0)[0] intercept
Total fit time: 0.737 seconds
Selected ARIMA Order: (0, 1, 1)
                               SARIMAX Results

==============================================================================
```

```
========
Dep. Variable:                MeanTemp   No. Observations:
143
Model:                 ARIMA(0, 1, 1)   Log Likelihood
41.070
Date:                Tue, 06 May 2025   AIC
-78.140
Time:                        14:34:56   BIC
-72.228
Sample:                             0   HQIC
-75.738
                                - 143

Covariance Type:                  opg

==============================================================================
========
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
--------
ma.L1         -0.6489      0.067     -9.684      0.000      -0.780
-0.518
sigma2         0.0327      0.004      7.340      0.000       0.024
0.041
==============================================================================
=============
Ljung-Box (L1) (Q):                   0.02   Jarque-Bera (JB):
1.60
Prob(Q):                              0.88   Prob(JB):
0.45
Heteroskedasticity (H):               1.49   Skew:
-0.20
Prob(H) (two-sided):                  0.18   Kurtosis:
2.66
==============================================================================
=============

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
```




```
Forecasted Annual Temperatures:
     Year  Min Forecast  Max Forecast  Mean Forecast
143  2023      8.734658     11.052061       9.905106
144  2024      8.734658     11.052061       9.905106
145  2025      8.734658     11.052061       9.905106
146  2026      8.734658     11.052061       9.905106
147  2027      8.734658     11.052061       9.905106
```

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_absolute_error, mean_squared_error


# Convert to datetime if not already
if 'Date' in df.columns:
    df['Date'] = pd.to_datetime(df['Date'])
    df.set_index('Date', inplace=True)

# Set year column for grouping
df['Year'] = df.index.year

# Initialize parameters
group_size = 10
start_year = 1880
end_year = 2020

# Function to perform ADF test
def adf_test(series):
    result = adfuller(series.dropna())
```

```
    return result[0], result[1]

# Store results
mae_rmse_arima_results = []

# Loop over 20-year groups
for start in range(start_year, end_year, group_size):
    df_group = df[(df['Year'] >= start) & (df['Year'] < start +
group_size)].copy()
    if len(df_group) < 40:  # skip very small samples
        continue

    # ADF Test
    _, pval = adf_test(df_group['Temperature'])
    if pval > 0.05:
        df_group['Temperature_Diff'] = df_group['Temperature'].diff()
        diff_data = df_group['Temperature_Diff'].dropna()
    else:
        diff_data = df_group['Temperature']

    # Fit ARIMA (5,1,2) arbitrarily; in full work, ACF/PACF should
guide p,d,q
    try:
        model = ARIMA(df_group['Temperature'], order=(5,1,2))
        arima_result = model.fit()

        # Forecast last 4 days
        forecast = arima_result.forecast(steps=4)
        y_actual = df_group['Temperature'][-4:]
        y_pred = forecast[:4]

        # Compute MAE and RMSE
        mae = mean_absolute_error(y_actual, y_pred)
        rmse = np.sqrt(mean_squared_error(y_actual, y_pred))
        mae_rmse_arima_results.append([f"{start}-{start +
group_size}", mae, rmse])
    except:
        continue

# Create results DataFrame
results_arima_df = pd.DataFrame(mae_rmse_arima_results, columns=["Time
Period", "ARIMA_MAE", "ARIMA_RMSE"])
results_arima_df


---------------------------------------------------------------------
-----
AttributeError                            Traceback (most recent call
last)
Cell In[22], line 17
```

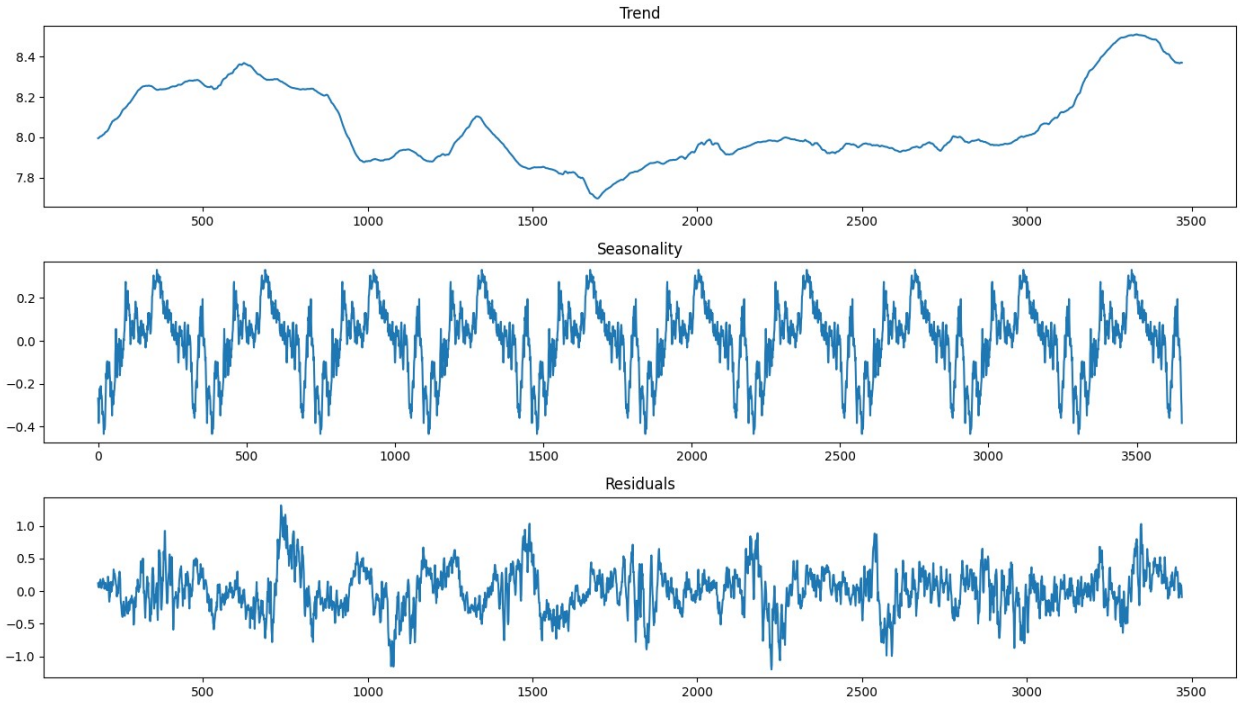
```
     14     df.set_index('Date', inplace=True)
     16 # Set year column for grouping
---> 17 df['Year'] = df.index.year
     19 # Initialize parameters
     20 group_size = 10

AttributeError: 'RangeIndex' object has no attribute 'year'
```

lstm models

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.seasonal import seasonal_decompose

from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

# Define group range
group_size = 10
start_year = 1880
end_year = 2020

# ADF Test
def adf_test(series):
    result = adfuller(series.dropna())
    print(f"ADF Statistic: {result[0]}")
    print(f"p-value: {result[1]}")
    if result[1] > 0.05:
        print("Non-stationary series (p > 0.05)")
    else:
        print("Stationary series (p <= 0.05)")

# Function to create sequences for LSTM
def create_sequences(data, seq_length=10):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:i + seq_length])
```

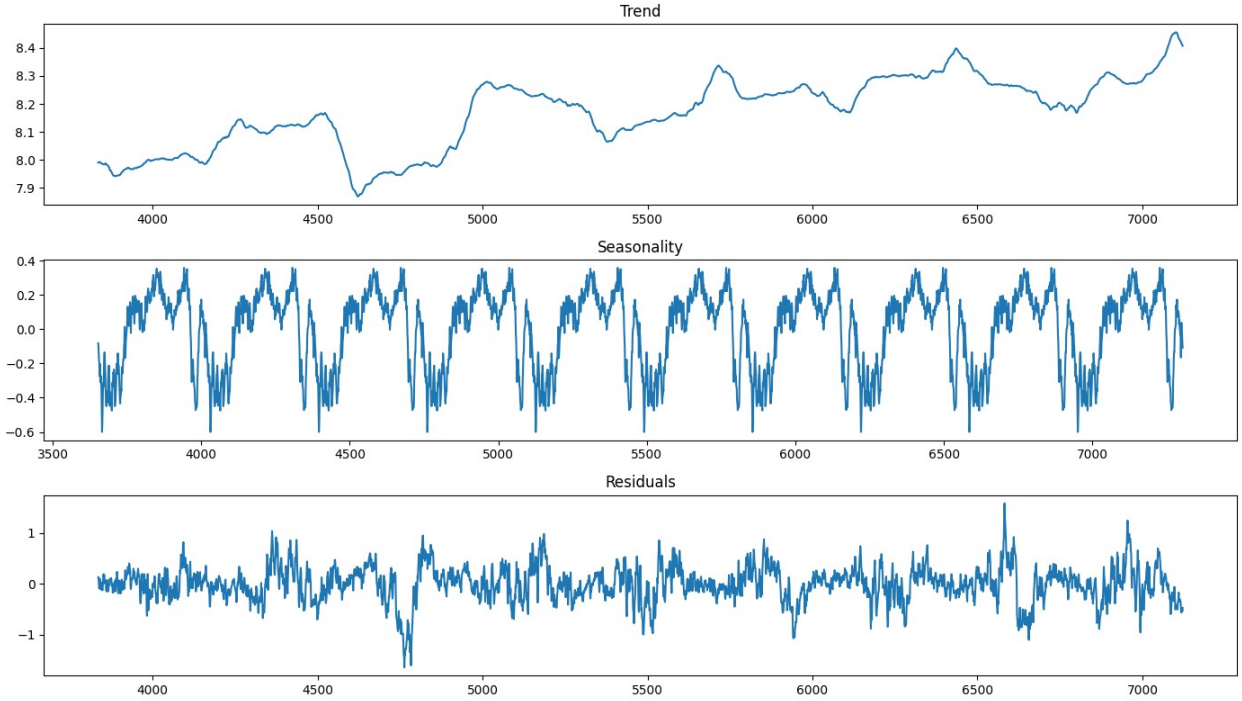
```
        y.append(data[i + seq_length])
    return np.array(X), np.array(y)

# Assuming df is already loaded and contains 'Year' and 'Temperature'
mae_rmse_results = []

for start in range(start_year, end_year, group_size):
    df_group = df[(df['Year'] >= start) & (df['Year'] < start +
group_size)].copy()
    print(f"\nAnalyzing period: {start}-{start + group_size}")

    # Decompose time series
    decomposition = seasonal_decompose(df_group['Temperature'],
model='additive', period=365)
    fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(14, 8))
    decomposition.trend.plot(ax=ax1, title='Trend')
    decomposition.seasonal.plot(ax=ax2, title='Seasonality')
    decomposition.resid.plot(ax=ax3, title='Residuals')
    plt.tight_layout()
    plt.show()

    # Normalize Data
    scaler = MinMaxScaler(feature_range=(0, 1))
    df_group['Temperature_Scaled'] =
scaler.fit_transform(df_group[['Temperature']])

    # Prepare Sequences for LSTM
    seq_length = 30  # Using 30 days of past data
    X, y = create_sequences(df_group['Temperature_Scaled'].values,
seq_length)

    # Reshape for LSTM
    X = X.reshape((X.shape[0], X.shape[1], 1))

    # Train-Test Split
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, shuffle=False)

    # Build LSTM Model
    model = Sequential([
        LSTM(50, activation='relu', return_sequences=True,
input_shape=(seq_length, 1)),
        LSTM(50, activation='relu'),
        Dense(1)
    ])
    model.compile(optimizer='adam', loss='mse')

    # Train Model
    model.fit(X_train, y_train, epochs=20, batch_size=16, verbose=1)
```

```
    # Predictions
    y_pred_lstm = model.predict(X_test)
    y_pred_lstm = scaler.inverse_transform(y_pred_lstm)

    # Evaluate Performance
    y_test_actual = scaler.inverse_transform(y_test.reshape(-1, 1))
    mae_lstm = mean_absolute_error(y_test_actual, y_pred_lstm)
    rmse_lstm = np.sqrt(mean_squared_error(y_test_actual,
y_pred_lstm))

    # Store Results
    mae_rmse_results.append([f"{start}-{start + group_size}",
mae_lstm, rmse_lstm])
    print(f'LSTM MAE ({start}-{start + group_size}): {mae_lstm:.4f},
RMSE: {rmse_lstm:.4f}')

# Create DataFrame for Results
results_df = pd.DataFrame(mae_rmse_results, columns=["Time Period",
"MAE", "RMSE"])
print("\nSummary of LSTM Performance:")
print(results_df)

# Plot Results
plt.figure(figsize=(12, 6))
plt.plot(results_df["Time Period"], results_df["MAE"], marker='o',
label="MAE")
plt.plot(results_df["Time Period"], results_df["RMSE"], marker='s',
label="RMSE")
plt.xlabel("Time Period")
plt.ylabel("Error Value")
plt.title("MAE and RMSE for Each 10-Year Group using LSTM")
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.show()


Analyzing period: 1880-1890
```



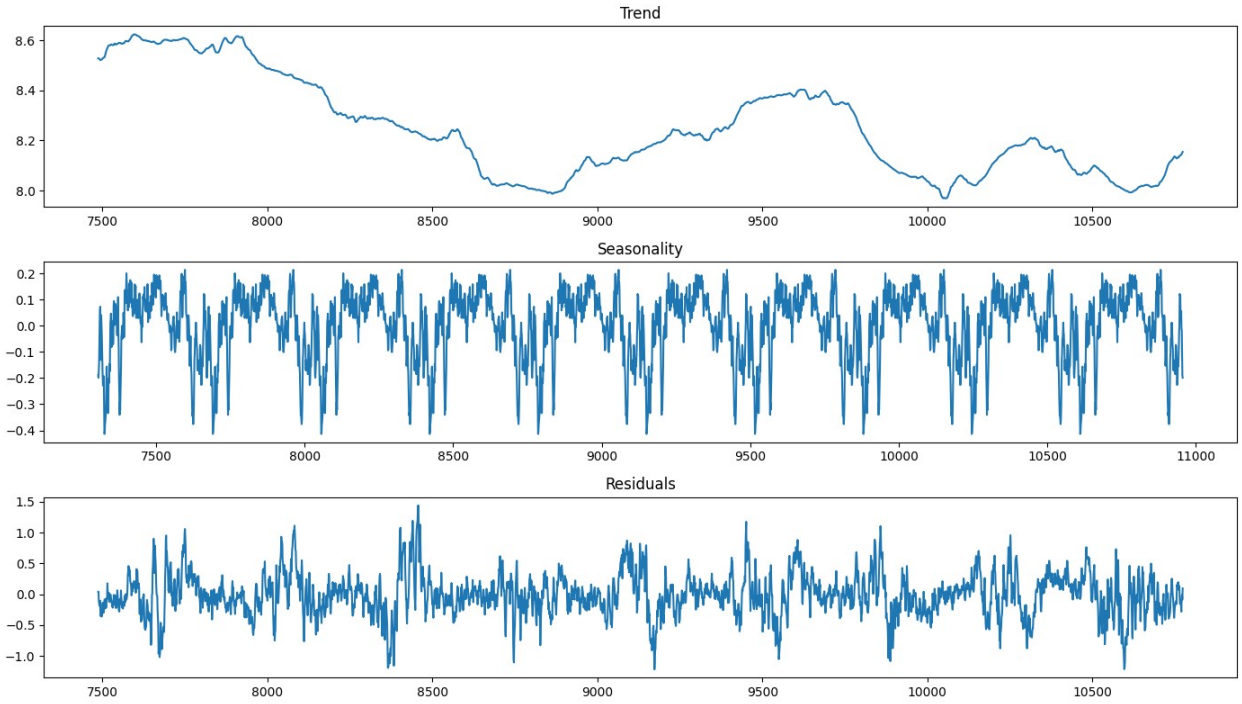

```
Epoch 1/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 5s 11ms/step - loss: 0.0464
Epoch 2/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0066
Epoch 3/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0063
Epoch 4/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0044
Epoch 5/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0032
Epoch 6/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0023
Epoch 7/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0020
Epoch 8/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0020
Epoch 9/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0021
Epoch 10/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0016
Epoch 11/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0019
Epoch 12/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0016
Epoch 13/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0016
Epoch 14/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0017
```

```
Epoch 15/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0017
Epoch 16/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0015
Epoch 17/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0016
Epoch 18/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - loss: 0.0016
Epoch 19/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0015
Epoch 20/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0015
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step
LSTM MAE (1880-1890): 0.1122, RMSE: 0.1435

Analyzing period: 1890-1900
```



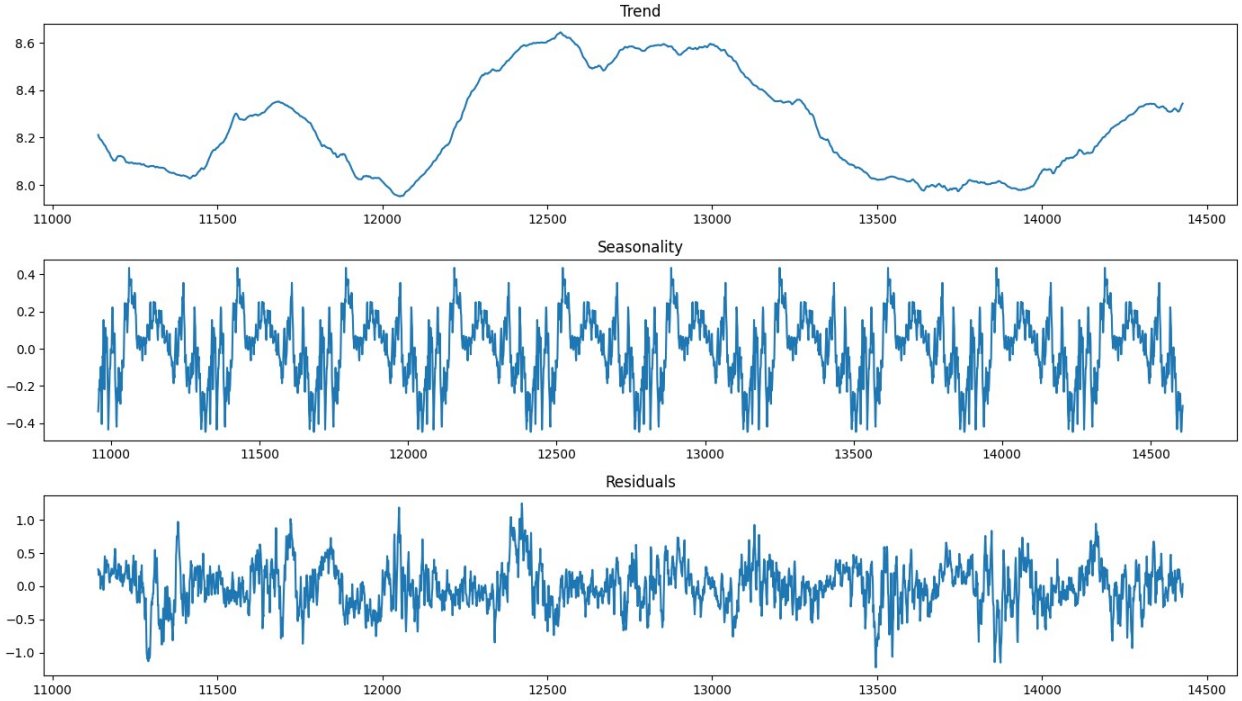

```
Epoch 1/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 4s 10ms/step - loss: 0.0894
Epoch 2/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0053
Epoch 3/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0045
Epoch 4/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0039
Epoch 5/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0030
```

```
Epoch 6/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0026
Epoch 7/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0022
Epoch 8/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0021
Epoch 9/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0015
Epoch 10/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0017
Epoch 11/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0016
Epoch 12/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0017
Epoch 13/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0015
Epoch 14/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0016
Epoch 15/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0016
Epoch 16/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0015
Epoch 17/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0015
Epoch 18/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0015
Epoch 19/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0020
Epoch 20/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0015
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step
LSTM MAE (1890-1900): 0.1150, RMSE: 0.1482

Analyzing period: 1900-1910
```



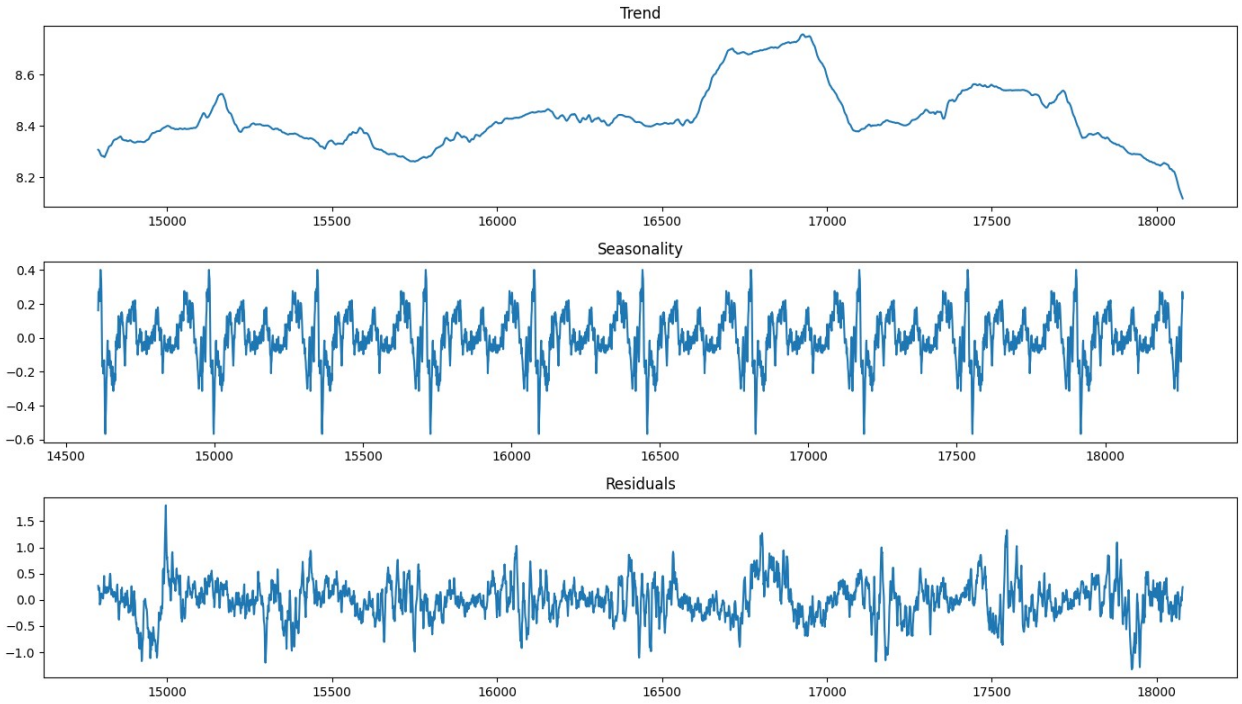

```
Epoch 1/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 4s 10ms/step - loss: 0.0802
Epoch 2/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0097
Epoch 3/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0072
Epoch 4/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0062
Epoch 5/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0046
Epoch 6/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0038
Epoch 7/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0038
Epoch 8/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0029
Epoch 9/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0025
Epoch 10/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0025
Epoch 11/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0026
Epoch 12/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0022
Epoch 13/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0022
Epoch 14/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0024
```

```
Epoch 15/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0024
Epoch 16/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0025
Epoch 17/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0027
Epoch 18/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0024
Epoch 19/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0024
Epoch 20/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0024
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step
LSTM MAE (1900-1910): 0.1270, RMSE: 0.1642

Analyzing period: 1910-1920
```



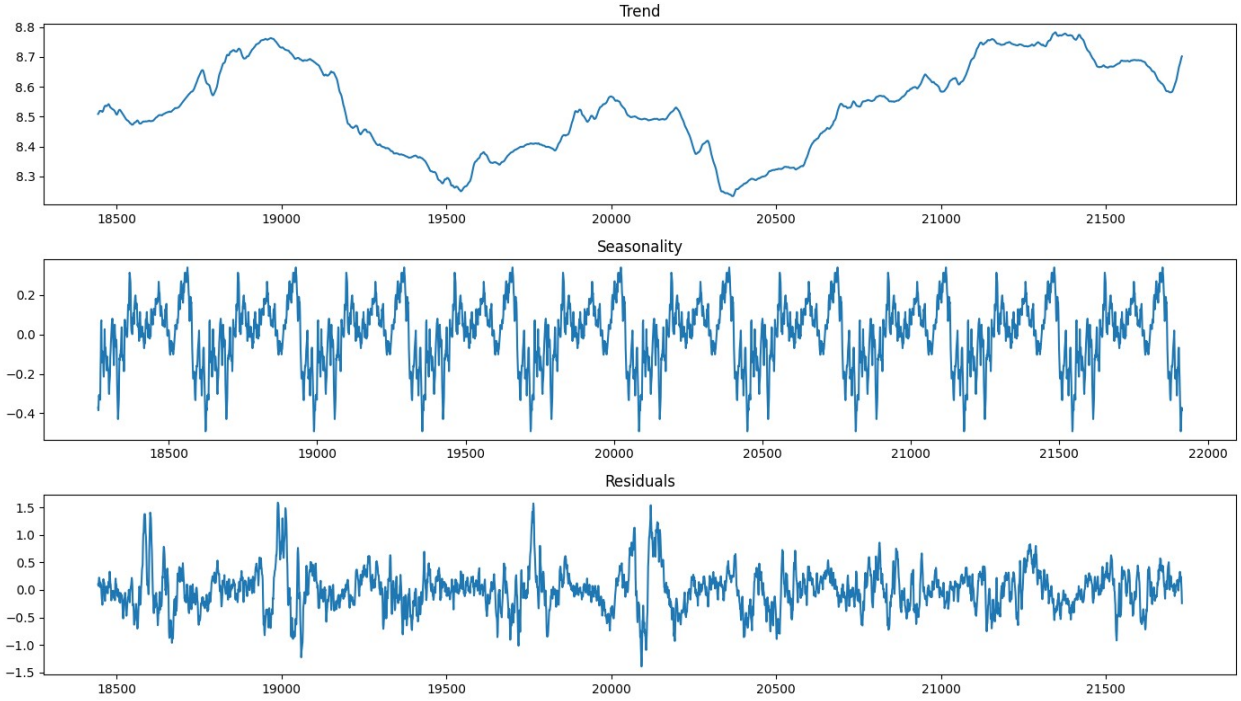

```
Epoch 1/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 4s 10ms/step - loss: 0.0872
Epoch 2/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0095
Epoch 3/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0086
Epoch 4/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0069
Epoch 5/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0063
```

```
Epoch 6/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0048
Epoch 7/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0031
Epoch 8/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0024
Epoch 9/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0022
Epoch 10/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0023
Epoch 11/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0020
Epoch 12/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0019
Epoch 13/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0019
Epoch 14/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0024
Epoch 15/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0020
Epoch 16/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0019
Epoch 17/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0023
Epoch 18/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0021
Epoch 19/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0021
Epoch 20/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0020
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step
LSTM MAE (1910-1920): 0.1140, RMSE: 0.1437

Analyzing period: 1920-1930
```



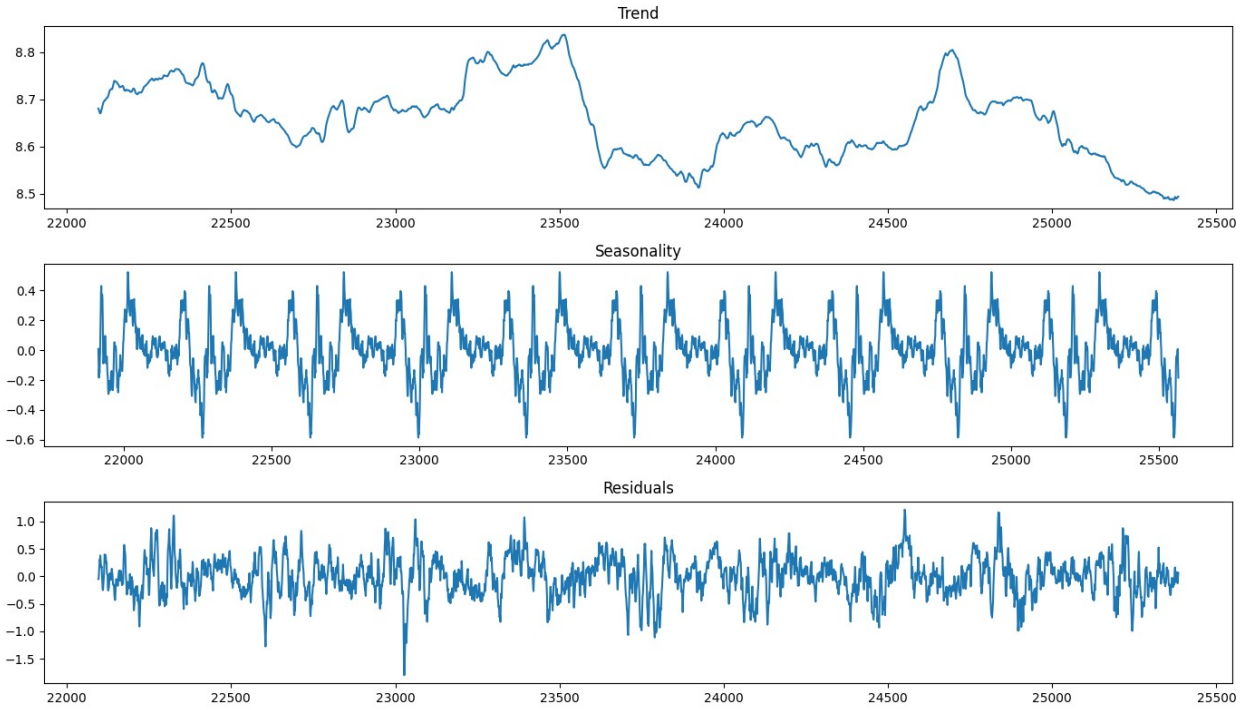

```
Epoch 1/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - loss: 0.0709
Epoch 2/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0073
Epoch 3/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0059
Epoch 4/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0042
Epoch 5/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0032
Epoch 6/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0024
Epoch 7/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0019
Epoch 8/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0022
Epoch 9/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0017
Epoch 10/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0017
Epoch 11/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0021
Epoch 12/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 13ms/step - loss: 0.0015
Epoch 13/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 13ms/step - loss: 0.0017
Epoch 14/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0015
```

```
Epoch 15/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0016
Epoch 16/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0015
Epoch 17/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0016
Epoch 18/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0016
Epoch 19/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0014
Epoch 20/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0016
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step
LSTM MAE (1920-1930): 0.1097, RMSE: 0.1429

Analyzing period: 1930-1940
```



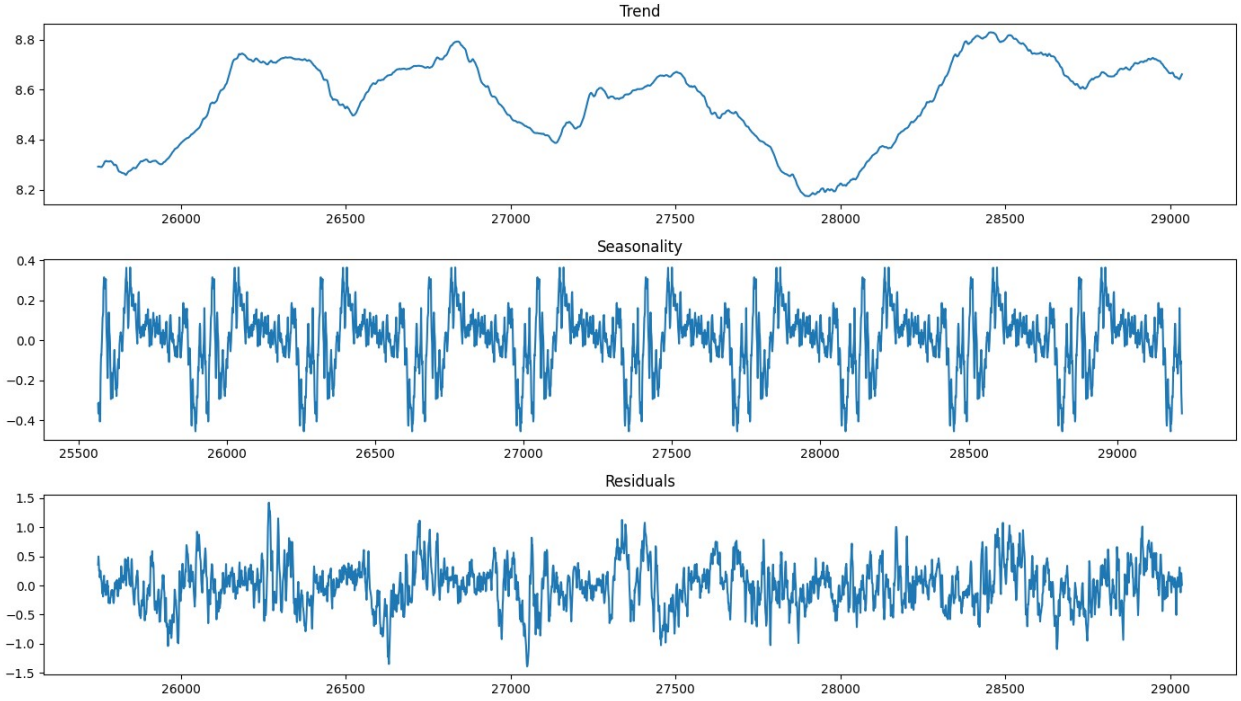

```
Epoch 1/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 5s 12ms/step - loss: 0.0624
Epoch 2/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 13ms/step - loss: 0.0102
Epoch 3/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0092
Epoch 4/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0062
Epoch 5/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0052
```

```
Epoch 6/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0041
Epoch 7/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0038
Epoch 8/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0024
Epoch 9/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0024
Epoch 10/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0020
Epoch 11/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0022
Epoch 12/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0023
Epoch 13/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0017
Epoch 14/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0017
Epoch 15/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0017
Epoch 16/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0017
Epoch 17/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0016
Epoch 18/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0015
Epoch 19/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0015
Epoch 20/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0016
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step
LSTM MAE (1930-1940): 0.0949, RMSE: 0.1206

Analyzing period: 1940-1950
```



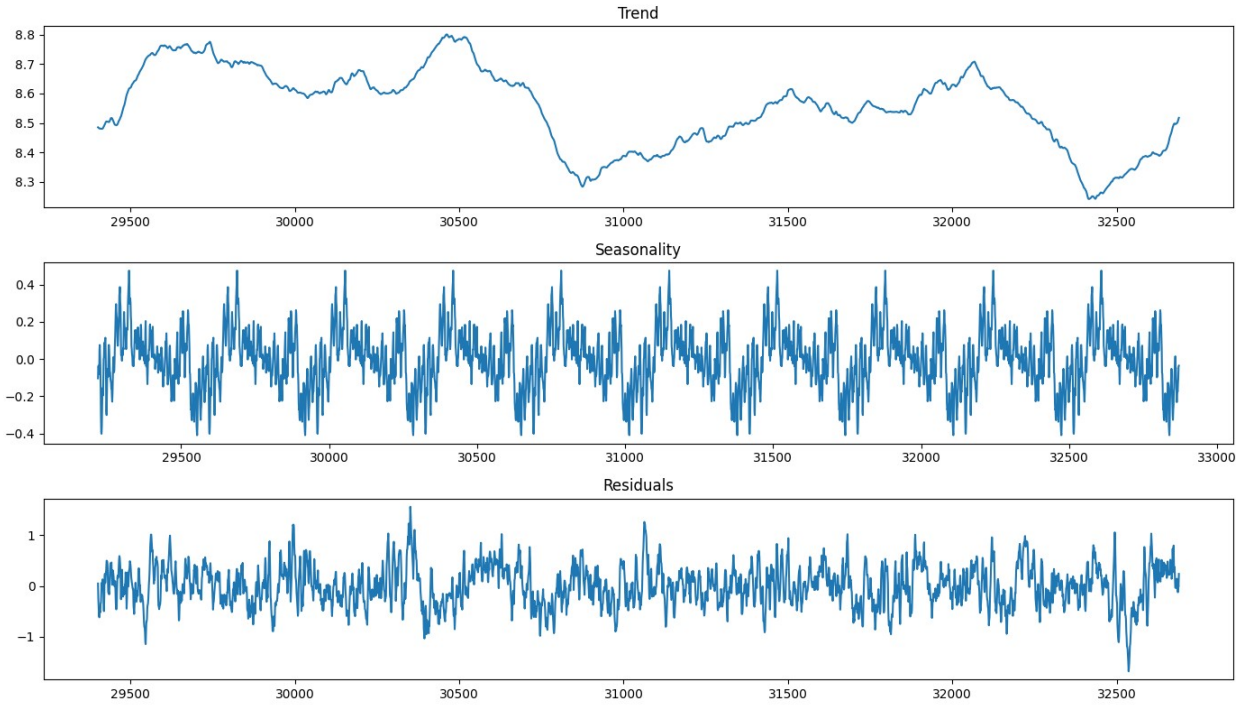

```
Epoch 1/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 4s 10ms/step - loss: 0.0917
Epoch 2/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0099
Epoch 3/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0078
Epoch 4/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0054
Epoch 5/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0041
Epoch 6/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0030
Epoch 7/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 3s 14ms/step - loss: 0.0026
Epoch 8/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0026
Epoch 9/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0022
Epoch 10/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0020
Epoch 11/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0018
Epoch 12/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0019
Epoch 13/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0023
Epoch 14/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 14ms/step - loss: 0.0018
```

```
Epoch 15/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0018
Epoch 16/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0018
Epoch 17/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0016
Epoch 18/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0016
Epoch 19/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0019
Epoch 20/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0019
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step
LSTM MAE (1940-1950): 0.1159, RMSE: 0.1465

Analyzing period: 1950-1960
```



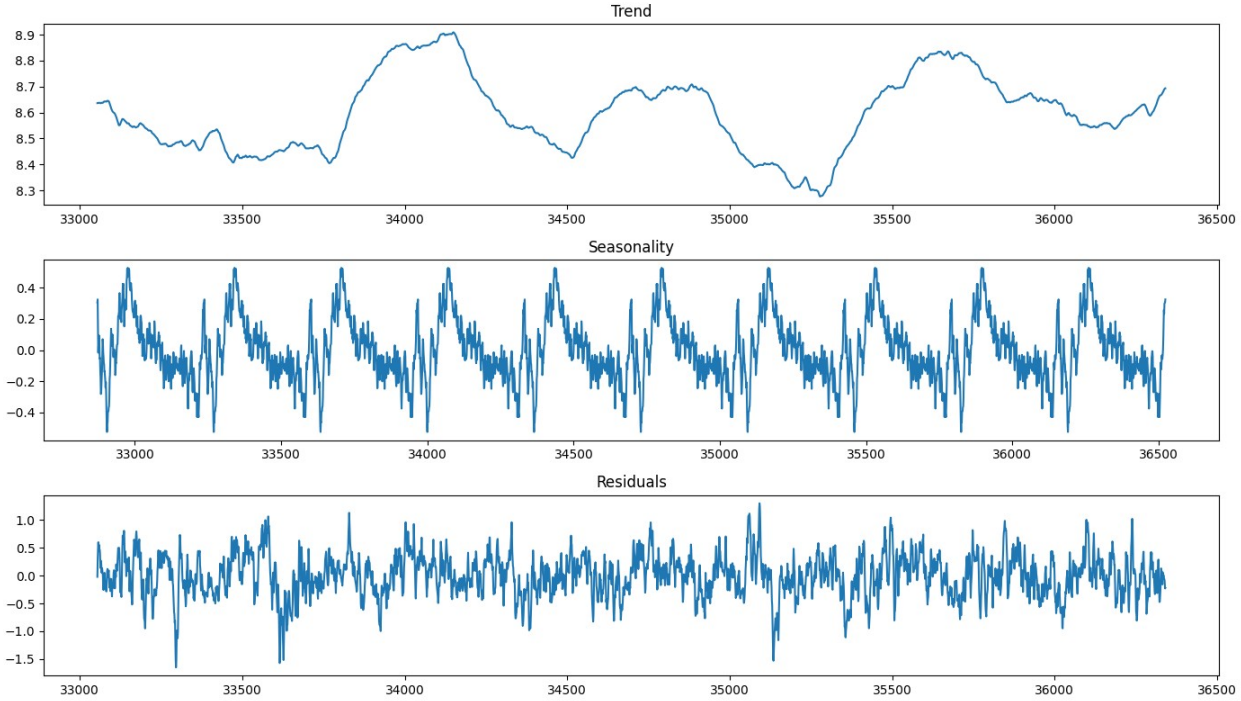

```
Epoch 1/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 4s 9ms/step - loss: 0.0692
Epoch 2/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0111
Epoch 3/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0076
Epoch 4/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0055
Epoch 5/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0045
```

```
Epoch 6/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0035
Epoch 7/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0027
Epoch 8/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0025
Epoch 9/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0022
Epoch 10/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0021
Epoch 11/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0022
Epoch 12/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0021
Epoch 13/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0021
Epoch 14/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0018
Epoch 15/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0019
Epoch 16/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0019
Epoch 17/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0019
Epoch 18/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0018
Epoch 19/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0021
Epoch 20/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0019
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step
LSTM MAE (1950-1960): 0.1342, RMSE: 0.1706

Analyzing period: 1960-1970
```



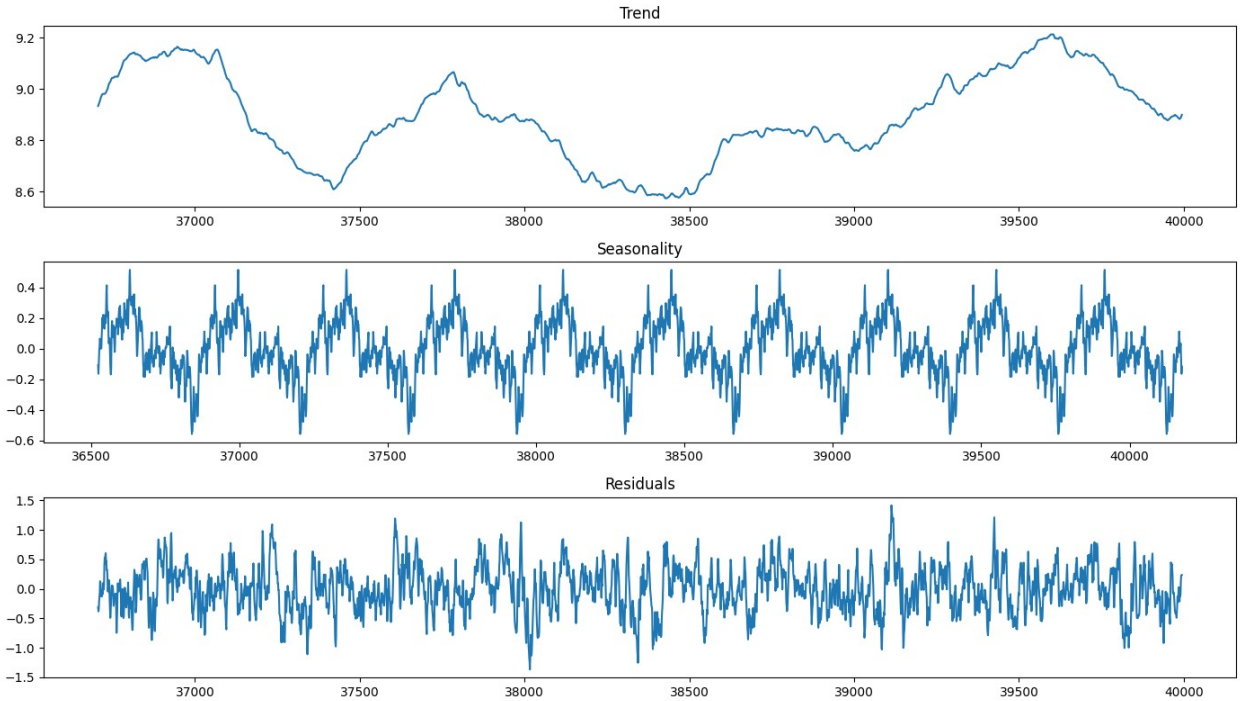

```
Epoch 1/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 4s 11ms/step - loss: 0.0446
Epoch 2/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0102
Epoch 3/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0090
Epoch 4/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0086
Epoch 5/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0076
Epoch 6/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0068
Epoch 7/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0048
Epoch 8/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0038
Epoch 9/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0031
Epoch 10/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0025
Epoch 11/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 13ms/step - loss: 0.0023
Epoch 12/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0019
Epoch 13/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0018
Epoch 14/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0018
```

```
Epoch 15/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0017
Epoch 16/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - loss: 0.0016
Epoch 17/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0017
Epoch 18/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0017
Epoch 19/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0016
Epoch 20/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0017
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step
LSTM MAE (1960-1970): 0.1244, RMSE: 0.1596

Analyzing period: 1970-1980
```



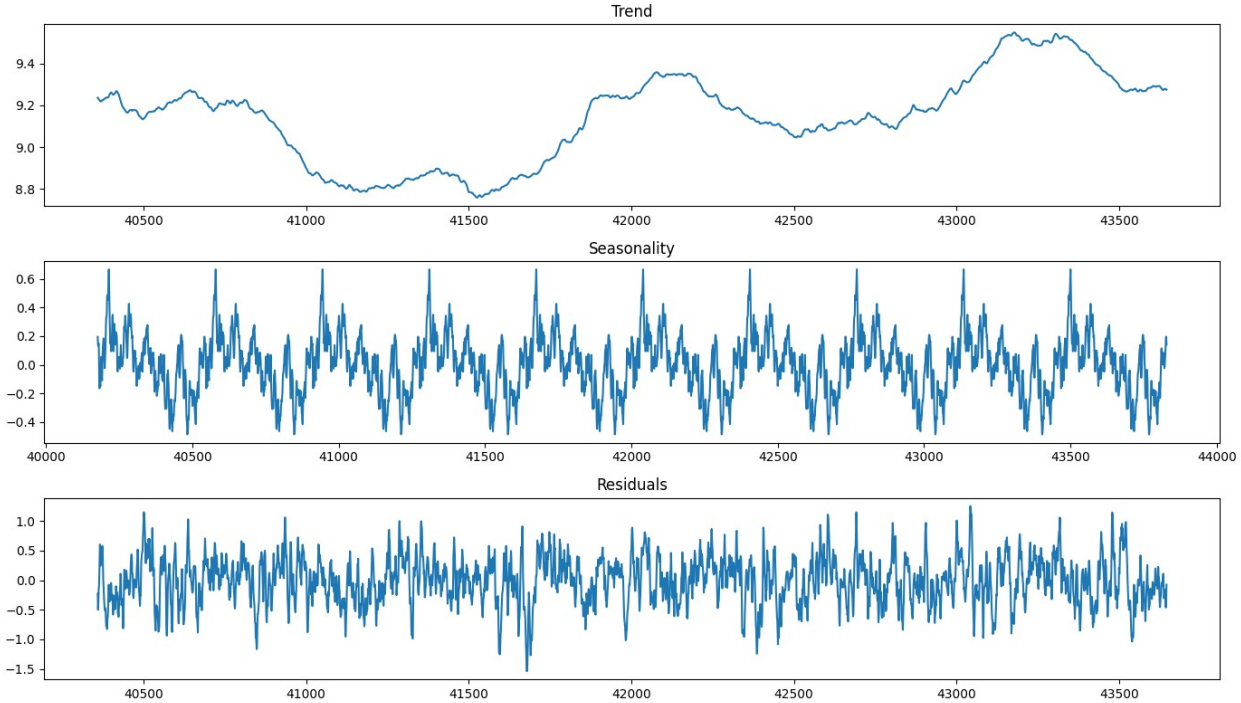

```
Epoch 1/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 4s 11ms/step - loss: 0.0824
Epoch 2/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0103
Epoch 3/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0076
Epoch 4/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0059
Epoch 5/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0042
```

```
Epoch 6/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0038
Epoch 7/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0031
Epoch 8/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0026
Epoch 9/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0020
Epoch 10/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0031
Epoch 11/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0024
Epoch 12/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0020
Epoch 13/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0020
Epoch 14/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0020
Epoch 15/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0022
Epoch 16/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0020
Epoch 17/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0022
Epoch 18/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0019
Epoch 19/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0021
Epoch 20/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0020
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step
LSTM MAE (1970-1980): 0.1208, RMSE: 0.1495

Analyzing period: 1980-1990
```



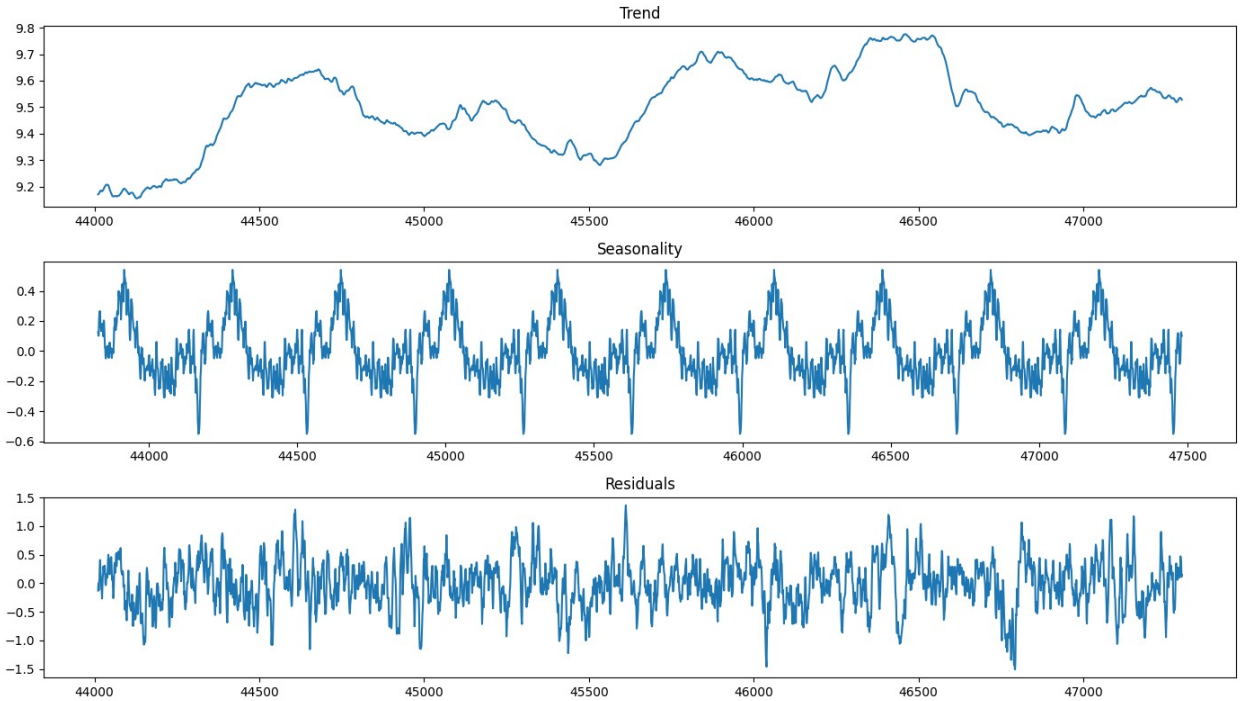

```
Epoch 1/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 4s 9ms/step - loss: 0.0805
Epoch 2/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0123
Epoch 3/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0077
Epoch 4/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0058
Epoch 5/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0036
Epoch 6/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0033
Epoch 7/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0033
Epoch 8/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0027
Epoch 9/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0029
Epoch 10/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0023
Epoch 11/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0021
Epoch 12/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0022
Epoch 13/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0023
Epoch 14/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0025
```

```
Epoch 15/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0023
Epoch 16/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0022
Epoch 17/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0022
Epoch 18/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0022
Epoch 19/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0023
Epoch 20/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0021
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step
LSTM MAE (1980-1990): 0.1397, RMSE: 0.1738

Analyzing period: 1990-2000
```



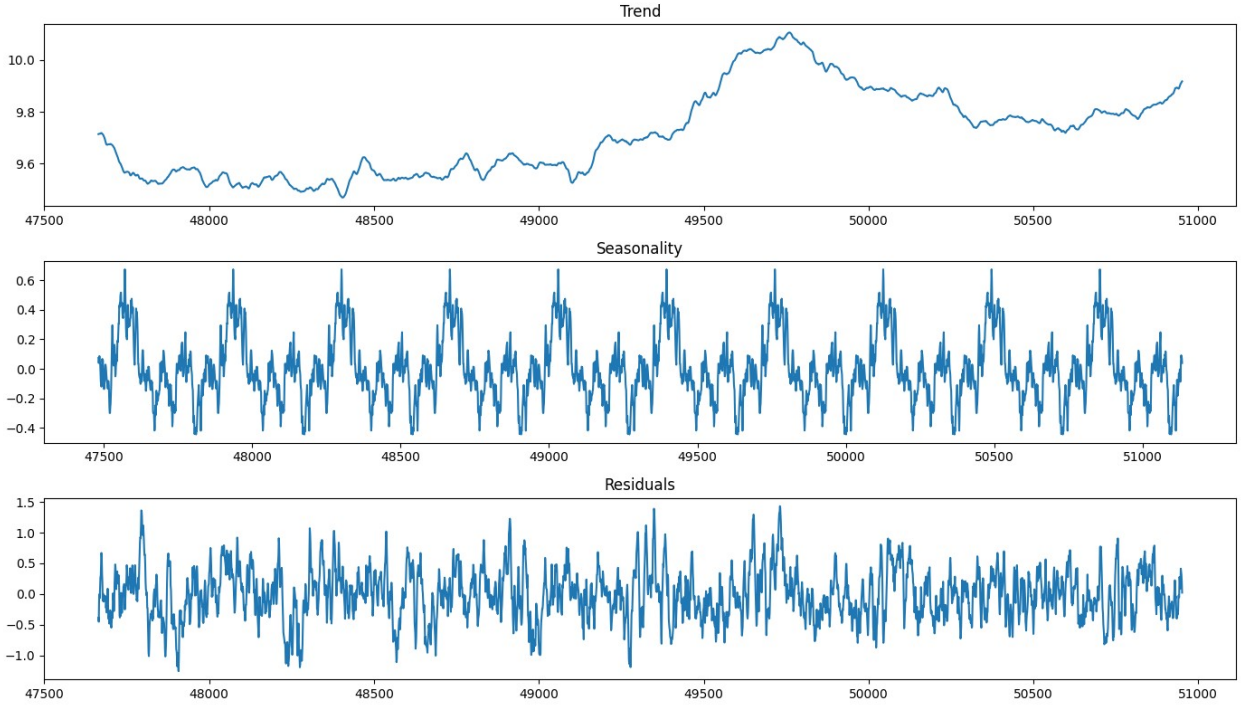

```
Epoch 1/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 4s 11ms/step - loss: 0.0741
Epoch 2/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0112
Epoch 3/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0089
Epoch 4/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0073
Epoch 5/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0048
```

```
Epoch 6/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0038
Epoch 7/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0032
Epoch 8/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0029
Epoch 9/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0026
Epoch 10/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0021
Epoch 11/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0024
Epoch 12/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0021
Epoch 13/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0020
Epoch 14/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0019
Epoch 15/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0019
Epoch 16/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0021
Epoch 17/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0028
Epoch 18/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0020
Epoch 19/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0019
Epoch 20/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0019
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step
LSTM MAE (1990-2000): 0.1223, RMSE: 0.1556

Analyzing period: 2000-2010
```



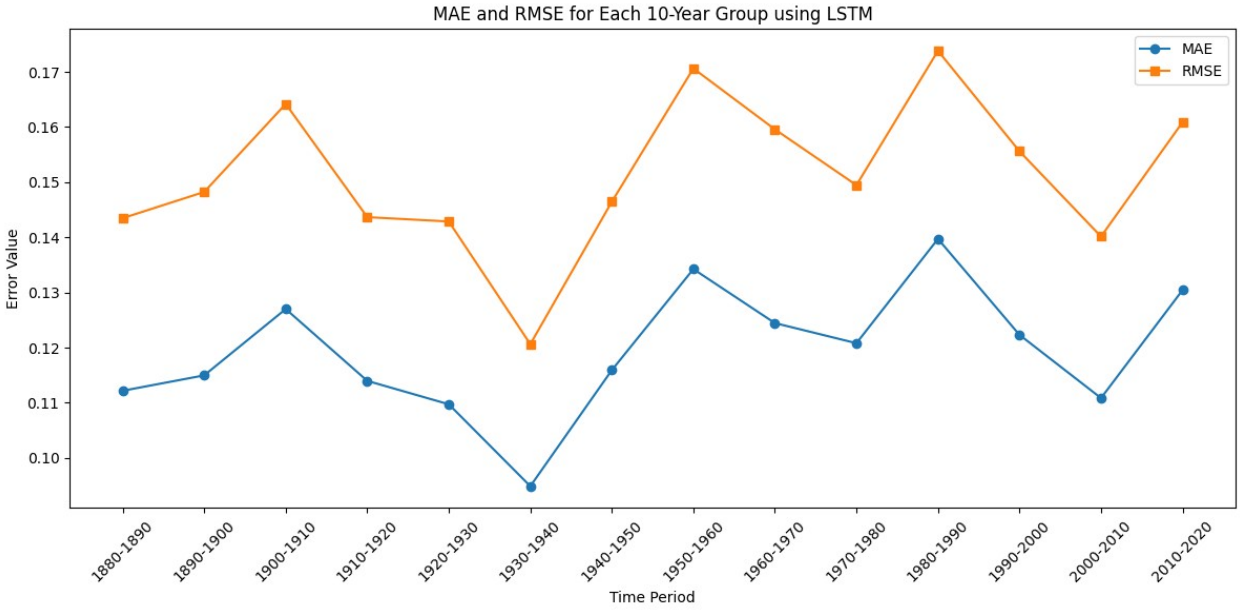

```
Epoch 1/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 4s 11ms/step - loss: 0.0526
Epoch 2/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0107
Epoch 3/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0090
Epoch 4/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0059
Epoch 5/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0042
Epoch 6/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0032
Epoch 7/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0028
Epoch 8/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0024
Epoch 9/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0021
Epoch 10/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 0.0022
Epoch 11/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0022
Epoch 12/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0021
Epoch 13/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0021
Epoch 14/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0021
```

```
Epoch 15/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - loss: 0.0021
Epoch 16/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - loss: 0.0022
Epoch 17/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0021
Epoch 18/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0021
Epoch 19/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0026
Epoch 20/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0021
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step
LSTM MAE (2000-2010): 0.1109, RMSE: 0.1401

Analyzing period: 2010-2020
```



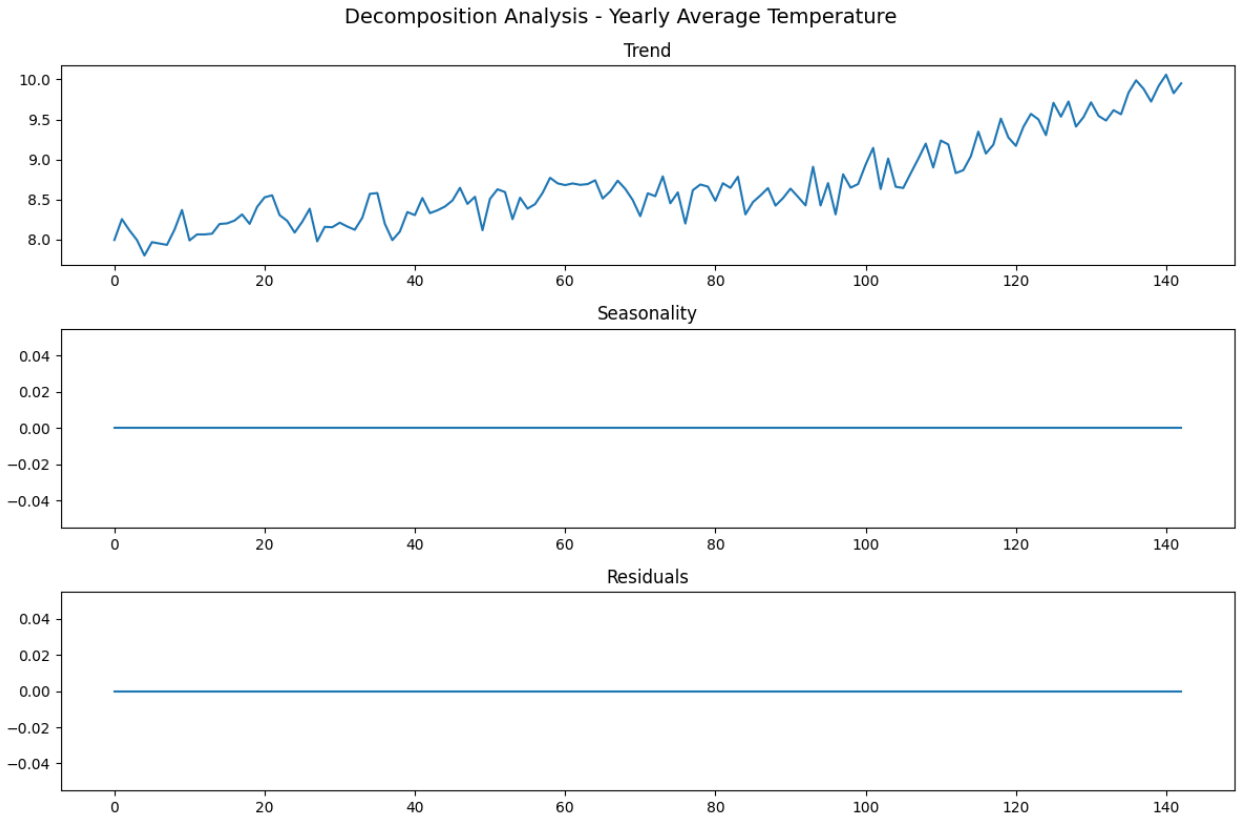

```
Epoch 1/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 4s 11ms/step - loss: 0.0759
Epoch 2/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0119
Epoch 3/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0085
Epoch 4/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0051
Epoch 5/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0033
```

```
Epoch 6/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0028
Epoch 7/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0040
Epoch 8/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0020
Epoch 9/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0025
Epoch 10/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0019
Epoch 11/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0022
Epoch 12/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0020
Epoch 13/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0020
Epoch 14/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0021
Epoch 15/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0021
Epoch 16/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0018
Epoch 17/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0019
Epoch 18/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0020
Epoch 19/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0019
Epoch 20/20
182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.0022
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step
LSTM MAE (2010-2020): 0.1304, RMSE: 0.1609

Summary of LSTM Performance:
   Time Period       MAE      RMSE
0    1880-1890  0.112176  0.143471
1    1890-1900  0.114968  0.148225
2    1900-1910  0.126996  0.164180
3    1910-1920  0.113969  0.143658
4    1920-1930  0.109726  0.142871
5    1930-1940  0.094874  0.120625
6    1940-1950  0.115922  0.146455
7    1950-1960  0.134238  0.170619
8    1960-1970  0.124446  0.159572
9    1970-1980  0.120801  0.149459
10   1980-1990  0.139709  0.173818
11   1990-2000  0.122313  0.155600
12   2000-2010  0.110852  0.140144
13   2010-2020  0.130430  0.160865
```




```
#yearly Avg

# Calculate yearly average
df_yearly = df.groupby('Year')['Temperature'].mean().reset_index()

# Calculate 5-year average
df['5YearGroup'] = (df['Year'] // 5) * 5
df_5year = df.groupby('5YearGroup')
['Temperature'].mean().reset_index()

from statsmodels.tsa.seasonal import seasonal_decompose

def train_lstm_on_series(series, label=""):
    # ---------- Decomposition ----------
    try:
        decomposition = seasonal_decompose(series, model='additive',
period=1)
        fig, axs = plt.subplots(3, 1, figsize=(12, 8))
        decomposition.trend.plot(ax=axs[0], title='Trend')
        decomposition.seasonal.plot(ax=axs[1], title='Seasonality')
        decomposition.resid.plot(ax=axs[2], title='Residuals')
        plt.suptitle(f"Decomposition Analysis - {label}", fontsize=14)
        plt.tight_layout()
        plt.show()
    except Exception as e:
        print(f"Decomposition failed: {e}")

    # ---------- Normalize ----------
    scaler = MinMaxScaler()
    series_scaled = scaler.fit_transform(series.values.reshape(-1, 1))
```

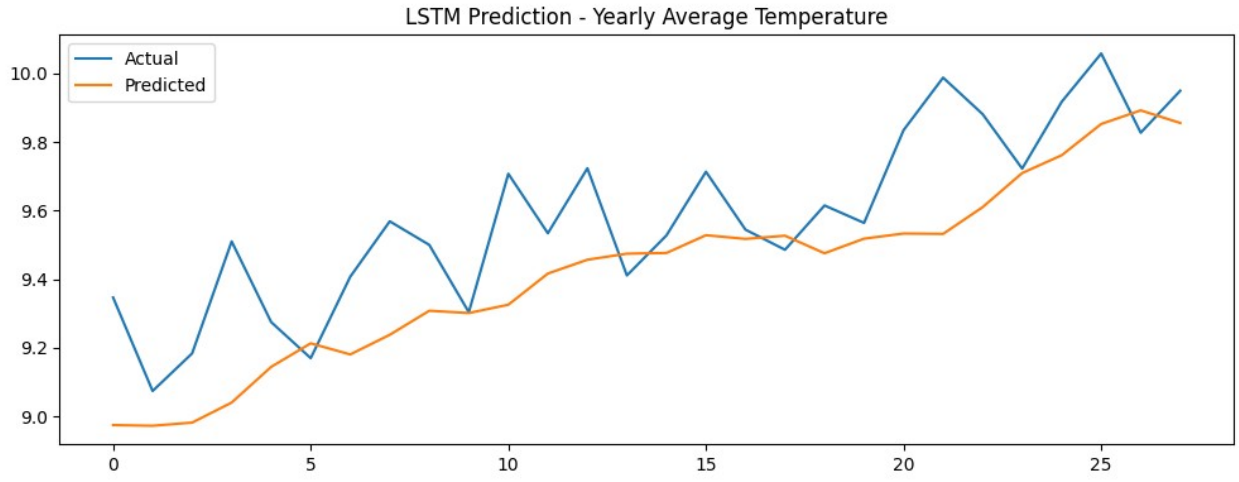
```
    # ---------- Create Sequences ----------
    seq_length = 5
    X, y = create_sequences(series_scaled, seq_length)
    X = X.reshape((X.shape[0], X.shape[1], 1))

    # ---------- Train-test Split ----------
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, shuffle=False)

    # ---------- Build and Train LSTM ----------
    model = Sequential([
        LSTM(50, activation='relu', return_sequences=True,
input_shape=(seq_length, 1)),
        LSTM(50, activation='relu'),
        Dense(1)
    ])
    model.compile(optimizer='adam', loss='mse')
    model.fit(X_train, y_train, epochs=50, batch_size=8, verbose=1)

    # ---------- Prediction ----------
    y_pred = model.predict(X_test)
    y_pred_inv = scaler.inverse_transform(y_pred)
    y_test_inv = scaler.inverse_transform(y_test.reshape(-1, 1))

    # ---------- Evaluation ----------
    mae = mean_absolute_error(y_test_inv, y_pred_inv)
    rmse = np.sqrt(mean_squared_error(y_test_inv, y_pred_inv))

    print(f"\nLSTM Performance ({label}):")
    print(f"MAE: {mae:.4f}")
    print(f"RMSE: {rmse:.4f}")

    # ---------- Plot Prediction ----------
    plt.figure(figsize=(10, 4))
    plt.plot(y_test_inv, label='Actual')
    plt.plot(y_pred_inv, label='Predicted')
    plt.title(f"LSTM Prediction - {label}")
    plt.legend()
    plt.tight_layout()
    plt.show()

    return model

# Yearly model
train_lstm_on_series(df_yearly['Temperature'], label="Yearly Average
Temperature")

# 5-Year model
train_lstm_on_series(df_5year['Temperature'], label="5-Year Average
Temperature")
```

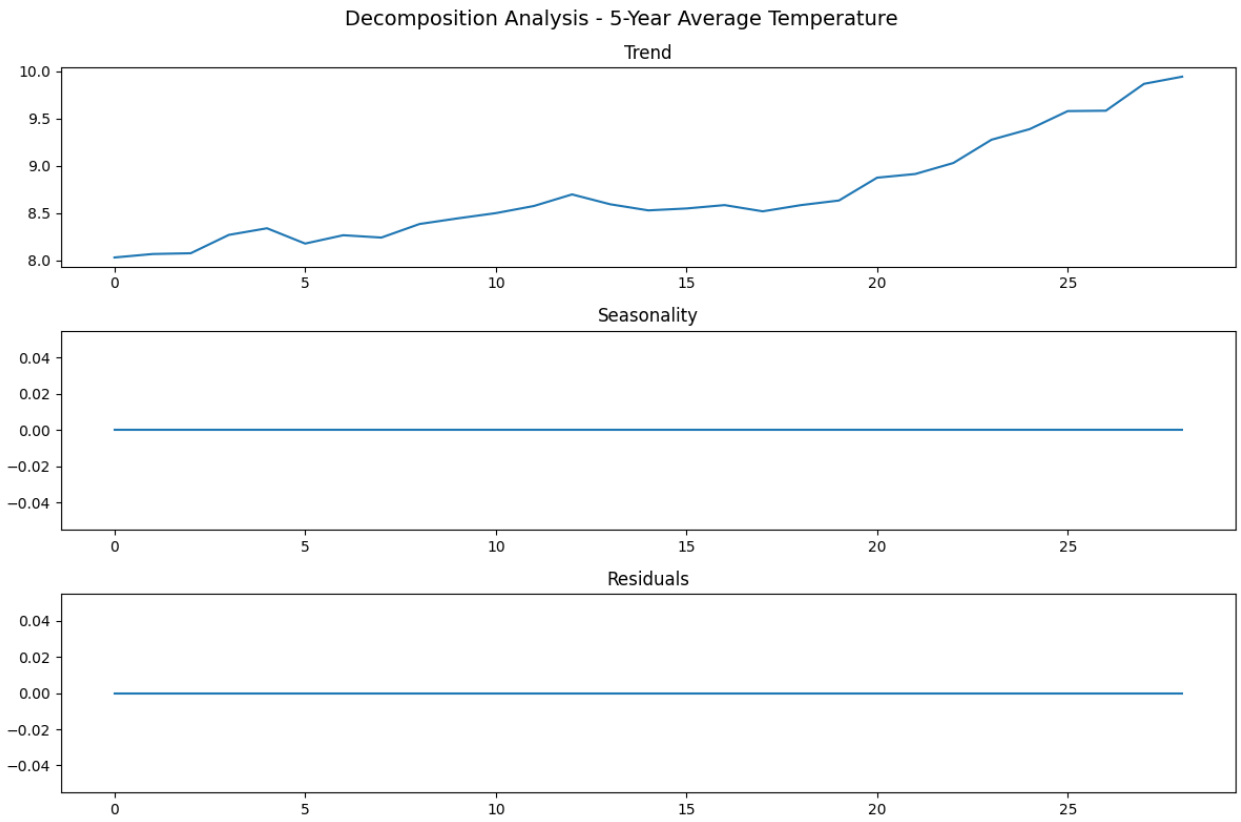



```
Epoch 1/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 5s 5ms/step - loss: 0.0979
Epoch 2/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0583
Epoch 3/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0172
Epoch 4/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0092
Epoch 5/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0100
Epoch 6/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0094
Epoch 7/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0087
Epoch 8/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0064
Epoch 9/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0077
Epoch 10/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0078
Epoch 11/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0065
Epoch 12/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0078
```

```
Epoch 13/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0073
Epoch 14/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0085
Epoch 15/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0087
Epoch 16/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0070
Epoch 17/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0070
Epoch 18/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0056
Epoch 19/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0066
Epoch 20/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0067
Epoch 21/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0069
Epoch 22/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0071
Epoch 23/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0064
Epoch 24/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0078
Epoch 25/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0061
Epoch 26/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0070
Epoch 27/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0071
Epoch 28/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0088
Epoch 29/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0070
Epoch 30/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0072
Epoch 31/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0052
Epoch 32/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0074
Epoch 33/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0076
Epoch 34/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0075
Epoch 35/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0064
Epoch 36/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0082
Epoch 37/50
```

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0069
Epoch 38/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0075
Epoch 39/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0061
Epoch 40/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0065
Epoch 41/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0078
Epoch 42/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0087
Epoch 43/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0068
Epoch 44/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0081
Epoch 45/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0064
Epoch 46/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0057
Epoch 47/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0067
Epoch 48/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0062
Epoch 49/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0068
Epoch 50/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0081
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 481ms/step

LSTM Performance (Yearly Average Temperature):
MAE: 0.1769
RMSE: 0.2216
```

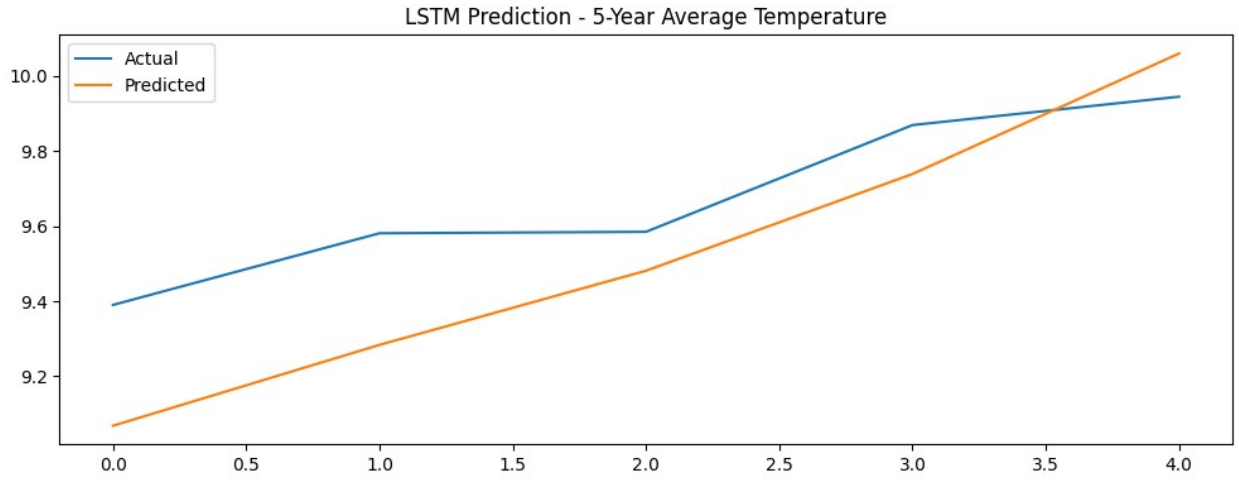





```
Epoch 1/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 5s 9ms/step - loss: 0.0973
Epoch 2/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0981
Epoch 3/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0842
Epoch 4/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0719
Epoch 5/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0699
Epoch 6/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0544
Epoch 7/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0480
Epoch 8/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0334
Epoch 9/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0260
Epoch 10/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0134
Epoch 11/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0130
Epoch 12/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0132
```

```
Epoch 13/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0125
Epoch 14/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0157
Epoch 15/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0131
Epoch 16/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0130
Epoch 17/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0096
Epoch 18/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0111
Epoch 19/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0144
Epoch 20/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0096
Epoch 21/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0101
Epoch 22/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0098
Epoch 23/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0121
Epoch 24/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0105
Epoch 25/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0103
Epoch 26/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0101
Epoch 27/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0096
Epoch 28/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0086
Epoch 29/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0111
Epoch 30/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0085
Epoch 31/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0086
Epoch 32/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0081
Epoch 33/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0084
Epoch 34/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0088
Epoch 35/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0093
Epoch 36/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0074
Epoch 37/50
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0084
Epoch 38/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0078
Epoch 39/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0090
Epoch 40/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0079
Epoch 41/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0081
Epoch 42/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0068
Epoch 43/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0068
Epoch 44/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0079
Epoch 45/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0057
Epoch 46/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0081
Epoch 47/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0066
Epoch 48/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0071
Epoch 49/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0065
Epoch 50/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0070
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 468ms/step

LSTM Performance (5-Year Average Temperature):
MAE: 0.1937
RMSE: 0.2159
```




```
<Sequential name=sequential_48, built=True>
```

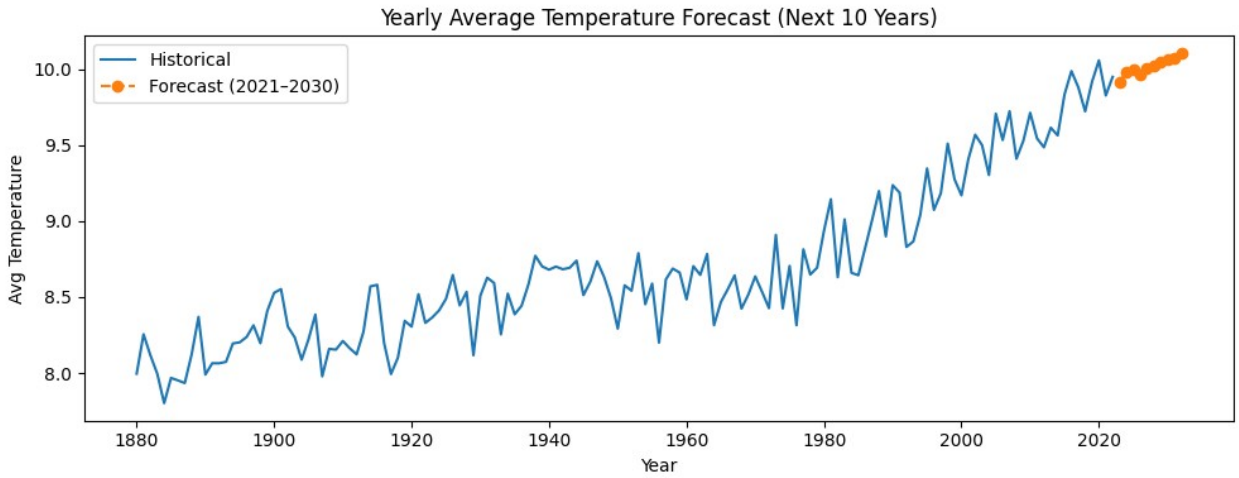
```
def forecast_future(model, series, years, scaler, seq_length=5):
    """Forecast future temperature values"""
    predictions = []
    input_seq = scaler.transform(series[-seq_length:].values.reshape(-
1, 1)).tolist()

    for _ in range(years):
        # Prepare input for prediction
        X_input = np.array(input_seq[-seq_length:]).reshape((1,
seq_length, 1))
        next_pred = model.predict(X_input, verbose=0)[0][0]
        predictions.append(next_pred)
        input_seq.append([next_pred])  # keep the sequence growing

    # Inverse scale predictions
    predictions_inv =
scaler.inverse_transform(np.array(predictions).reshape(-1,
1)).flatten()
    return predictions_inv

# Train on full data for forecasting
scaler = MinMaxScaler()
series_scaled =
scaler.fit_transform(df_yearly['Temperature'].values.reshape(-1, 1))

# Re-train model on entire dataset
X_full, y_full = create_sequences(series_scaled, seq_length=5)
X_full = X_full.reshape((X_full.shape[0], X_full.shape[1], 1))
model_full = Sequential([
    LSTM(50, activation='relu', return_sequences=True, input_shape=(5,
1)),
    LSTM(50, activation='relu'),
    Dense(1)
])
model_full.compile(optimizer='adam', loss='mse')
model_full.fit(X_full, y_full, epochs=50, batch_size=8, verbose=1)

# Forecast next 10 years
future_years = list(range(df_yearly['Year'].iloc[-1] + 1,
df_yearly['Year'].iloc[-1] + 11))
future_predictions = forecast_future(model_full,
df_yearly['Temperature'], 10, scaler)

# Plot
plt.figure(figsize=(10, 4))
plt.plot(df_yearly['Year'], df_yearly['Temperature'],
label='Historical')
plt.plot(future_years, future_predictions, label='Forecast (2021–
2030)', linestyle='--', marker='o')
plt.xlabel("Year")
```

```
plt.ylabel("Avg Temperature")
plt.title("Yearly Average Temperature Forecast (Next 10 Years)")
plt.legend()
plt.tight_layout()
plt.show()

# Print values
forecast_df = pd.DataFrame({'Year': future_years, 'Predicted
Temperature': future_predictions})
print("\nForecasted Temperatures (2021–2030):")
print(forecast_df)
```

```
Epoch 1/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 0.2090
Epoch 2/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.1328
Epoch 3/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0302
Epoch 4/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0153
Epoch 5/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0102
Epoch 6/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0076
Epoch 7/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0076
Epoch 8/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0072
Epoch 9/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0077
Epoch 10/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0070
Epoch 11/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0077
Epoch 12/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0076
Epoch 13/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0068
Epoch 14/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0072
Epoch 15/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0064
Epoch 16/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0070
Epoch 17/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0075
Epoch 18/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0069
Epoch 19/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0084
```

```
Epoch 20/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0057
Epoch 21/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0064
Epoch 22/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0066
Epoch 23/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0082
Epoch 24/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0082
Epoch 25/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0070
Epoch 26/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0074
Epoch 27/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0072
Epoch 28/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0077
Epoch 29/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0072
Epoch 30/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0075
Epoch 31/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0070
Epoch 32/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0065
Epoch 33/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0061
Epoch 34/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0060
Epoch 35/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0077
Epoch 36/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0065
Epoch 37/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0065
Epoch 38/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0068
Epoch 39/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0080
Epoch 40/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0082
Epoch 41/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0046
Epoch 42/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0067
Epoch 43/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0067
Epoch 44/50
```

```
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0075
Epoch 45/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0063
Epoch 46/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0081
Epoch 47/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0073
Epoch 48/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0067
Epoch 49/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0070
Epoch 50/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0069
WARNING:tensorflow:5 out of the last 26 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_dist
ributed at 0x000001192C49E980> triggered tf.function retracing.
Tracing is expensive and the excessive number of tracings could be due
to (1) creating @tf.function repeatedly in a loop, (2) passing tensors
with different shapes, (3) passing Python objects instead of tensors.
For (1), please define your @tf.function outside of the loop. For (2),
@tf.function has reduce_retracing=True option that can avoid
unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more
details.
```

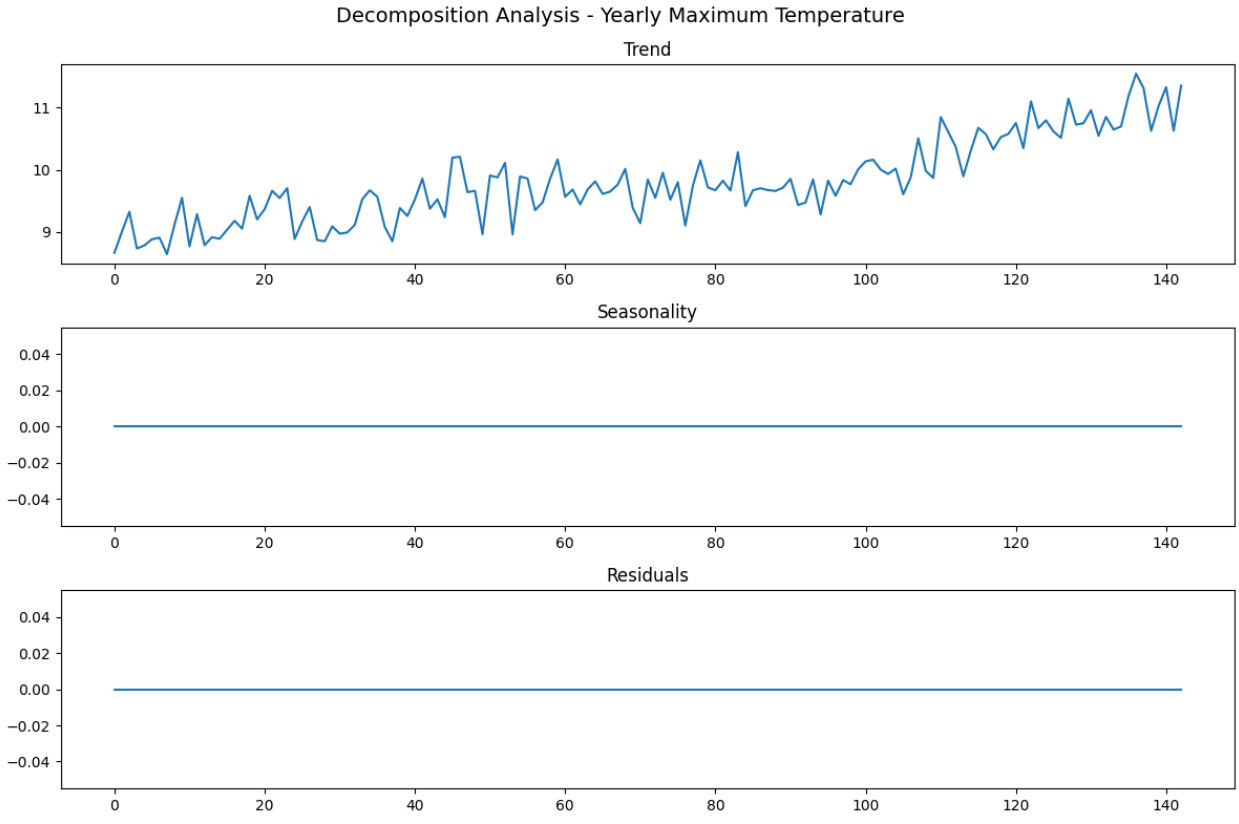



```
Forecasted Temperatures (2021–2030):
   Year  Predicted Temperature
0  2023               9.916318
1  2024               9.985366
2  2025               9.995470
3  2026               9.963815
4  2027              10.008984
```

```
5  2028              10.019996
6  2029              10.049272
7  2030              10.060635
8  2031              10.074349
9  2032              10.104771
```

```
#maximum


df_yearly_max = df.groupby('Year')['Temperature'].max().reset_index()

# Calculate 5-year average
df['5YearGroup'] = (df['Year'] // 5) * 5
df_5year_max = df.groupby('5YearGroup')
['Temperature'].max().reset_index()

def train_lstm_on_series(series, label=""):
    # Normalize
    scaler = MinMaxScaler()
    series_scaled = scaler.fit_transform(series.values.reshape(-1, 1))

    # Create sequences
    seq_length = 5
    X, y = create_sequences(series_scaled, seq_length)
    X = X.reshape((X.shape[0], X.shape[1], 1))

    # Train-test split
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, shuffle=False)

    # Model
    model = Sequential([
        LSTM(50, activation='relu', return_sequences=True,
input_shape=(seq_length, 1)),
        LSTM(50, activation='relu'),
        Dense(1)
    ])
    model.compile(optimizer='adam', loss='mse')
    model.fit(X_train, y_train, epochs=50, batch_size=8, verbose=1)

    # Predict and inverse transform
    y_pred = model.predict(X_test)
    y_pred_inv = scaler.inverse_transform(y_pred)
    y_test_inv = scaler.inverse_transform(y_test.reshape(-1, 1))

    # Metrics
    mae = mean_absolute_error(y_test_inv, y_pred_inv)
    rmse = np.sqrt(mean_squared_error(y_test_inv, y_pred_inv))

    print(f"\nLSTM Performance ({label}):")
    print(f"MAE: {mae:.4f}")
```

```
    print(f"RMSE: {rmse:.4f}")

    # Plot
    plt.figure(figsize=(10, 4))
    plt.plot(y_test_inv, label='Actual')
    plt.plot(y_pred_inv, label='Predicted')
    plt.title(f"LSTM Prediction - {label}")
    plt.legend()
    plt.tight_layout()
    plt.show()

    return model

# Yearly model
train_lstm_on_series(df_yearly_max['Temperature'], label="Yearly
Maximum Temperature")

# 5-Year model
train_lstm_on_series(df_5year_max['Temperature'], label="5-Year
Maximum Temperature")
```

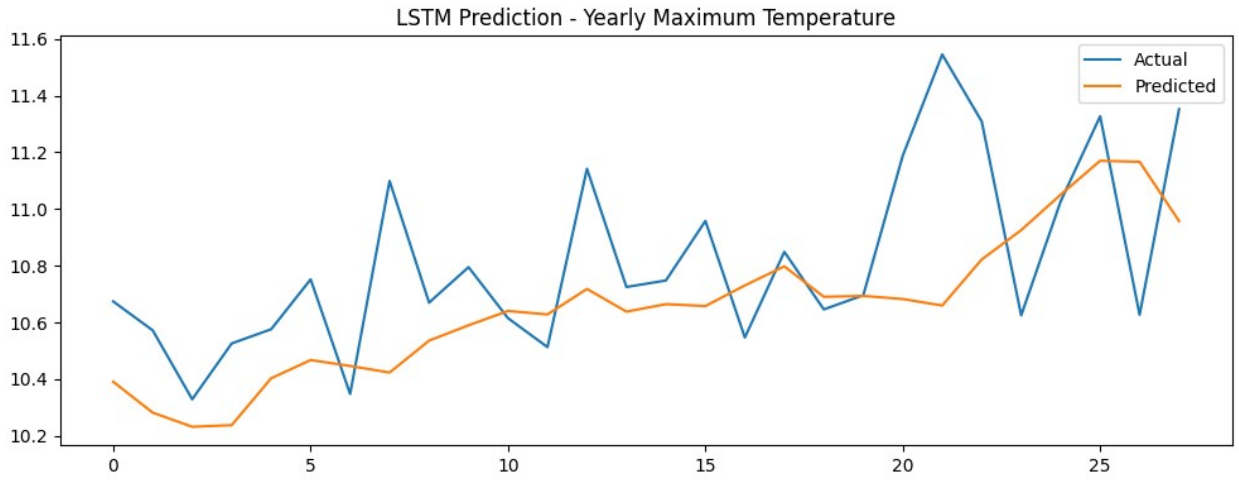


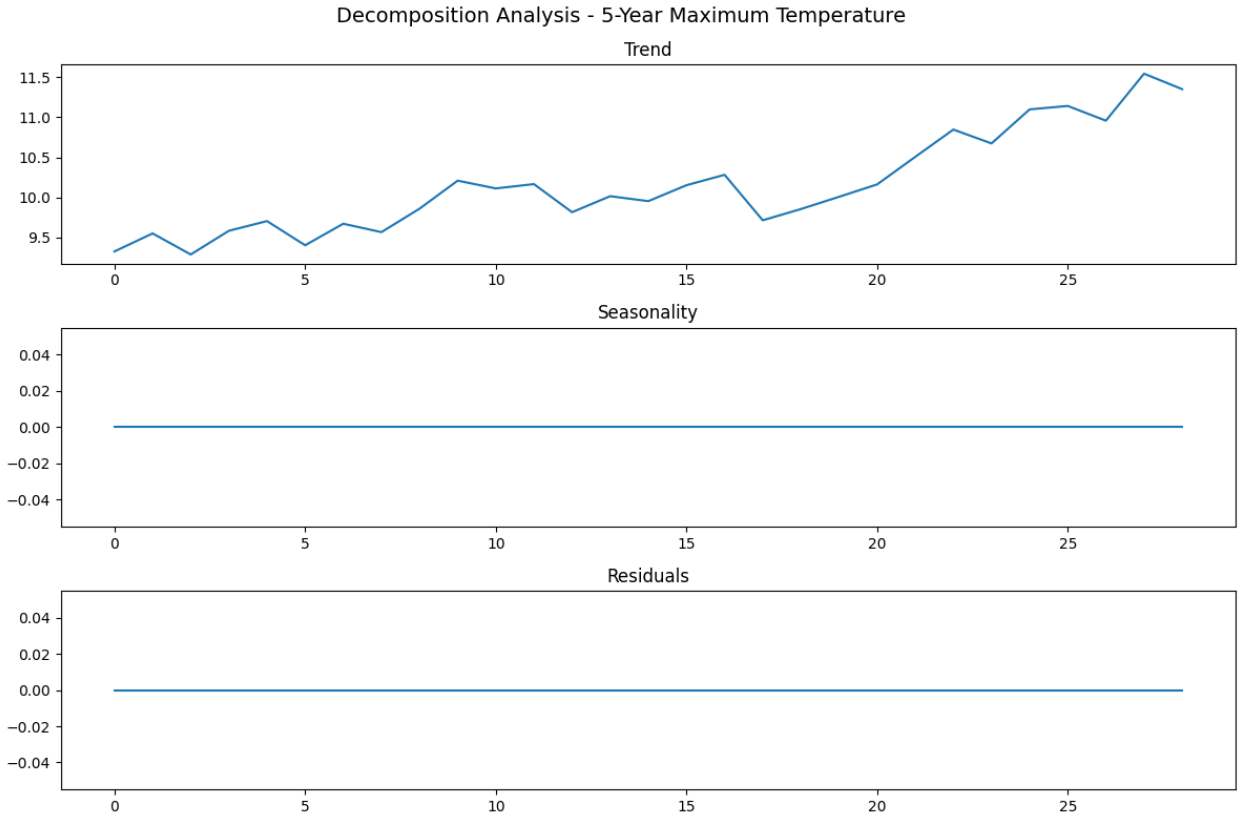

```
Epoch 1/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 4s 5ms/step - loss: 0.1002
Epoch 2/50
```

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0487
Epoch 3/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0214
Epoch 4/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0155
Epoch 5/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0177
Epoch 6/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0170
Epoch 7/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0188
Epoch 8/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0139
Epoch 9/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0143
Epoch 10/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0130
Epoch 11/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0120
Epoch 12/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0131
Epoch 13/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0117
Epoch 14/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0138
Epoch 15/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0115
Epoch 16/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0113
Epoch 17/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0149
Epoch 18/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0132
Epoch 19/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0108
Epoch 20/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0144
Epoch 21/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0115
Epoch 22/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0138
Epoch 23/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0132
Epoch 24/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0134
Epoch 25/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0151
Epoch 26/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0115
```

```
Epoch 27/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0147
Epoch 28/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0130
Epoch 29/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0110
Epoch 30/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0121
Epoch 31/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0121
Epoch 32/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0134
Epoch 33/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0130
Epoch 34/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0146
Epoch 35/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0117
Epoch 36/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0128
Epoch 37/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0115
Epoch 38/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0116
Epoch 39/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0122
Epoch 40/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0101
Epoch 41/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0119
Epoch 42/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0119
Epoch 43/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0121
Epoch 44/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0118
Epoch 45/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0149
Epoch 46/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0139
Epoch 47/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0128
Epoch 48/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0119
Epoch 49/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0135
Epoch 50/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0143
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 500ms/step
```

```
LSTM Performance (Yearly Maximum Temperature):
MAE: 0.2550
RMSE: 0.3311
```



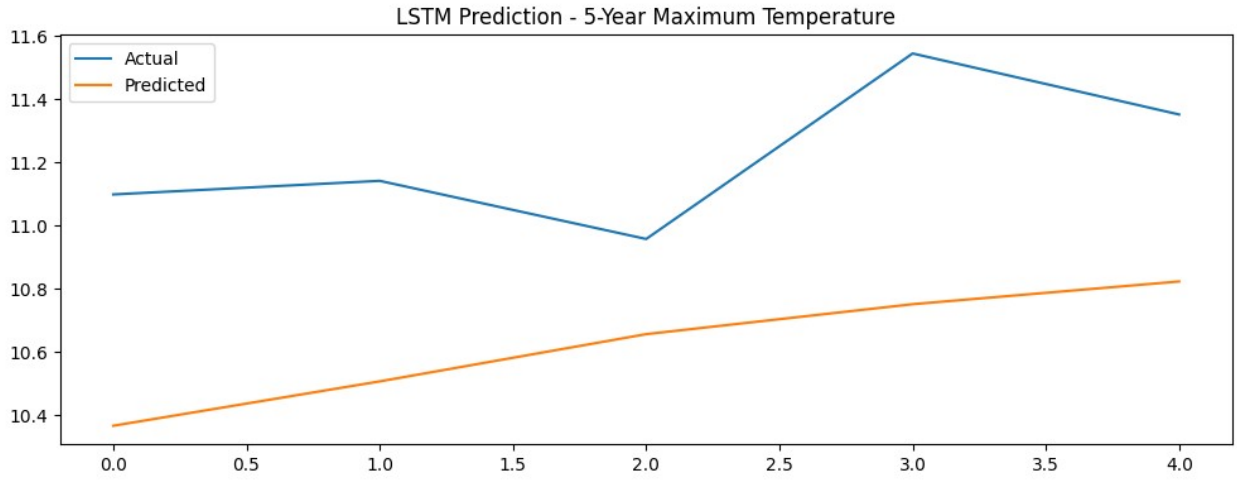



```
Epoch 1/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 5s 6ms/step - loss: 0.1466
Epoch 2/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.1295
```

```
Epoch 3/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.1180
Epoch 4/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.1011
Epoch 5/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.1012
Epoch 6/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0860
Epoch 7/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0713
Epoch 8/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0573
Epoch 9/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0433
Epoch 10/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0354
Epoch 11/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0292
Epoch 12/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0193
Epoch 13/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0203
Epoch 14/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0217
Epoch 15/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0234
Epoch 16/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0185
Epoch 17/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0196
Epoch 18/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0196
Epoch 19/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0199
Epoch 20/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0192
Epoch 21/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0221
Epoch 22/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0209
Epoch 23/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0182
Epoch 24/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0204
Epoch 25/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0229
Epoch 26/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0195
Epoch 27/50
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0182
Epoch 28/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0176
Epoch 29/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0235
Epoch 30/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0233
Epoch 31/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0200
Epoch 32/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0170
Epoch 33/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0202
Epoch 34/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0181
Epoch 35/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0200
Epoch 36/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0169
Epoch 37/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0157
Epoch 38/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0172
Epoch 39/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0209
Epoch 40/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0149
Epoch 41/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0190
Epoch 42/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0202
Epoch 43/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0169
Epoch 44/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0176
Epoch 45/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0192
Epoch 46/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0177
Epoch 47/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0195
Epoch 48/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0185
Epoch 49/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0182
Epoch 50/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0214
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 394ms/step
```

```
LSTM Performance (5-Year Maximum Temperature):
MAE: 0.5979
RMSE: 0.6225
```




```
<Sequential name=sequential_50, built=True>
```

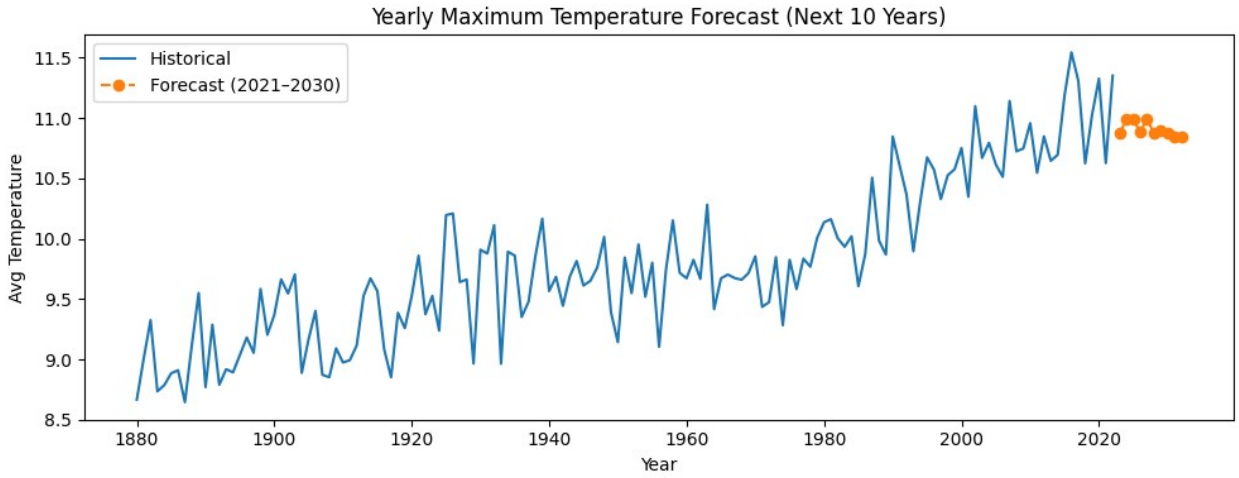
```
# Train on full data for forecasting
scaler = MinMaxScaler()
series_scaled =
scaler.fit_transform(df_yearly_max['Temperature'].values.reshape(-1,
1))

# Re-train model on entire dataset
X_full, y_full = create_sequences(series_scaled, seq_length=5)
X_full = X_full.reshape((X_full.shape[0], X_full.shape[1], 1))
model_full = Sequential([
    LSTM(50, activation='relu', return_sequences=True, input_shape=(5,
1)),
    LSTM(50, activation='relu'),
    Dense(1)
])
model_full.compile(optimizer='adam', loss='mse')
model_full.fit(X_full, y_full, epochs=50, batch_size=8, verbose=1)

# Forecast next 10 years
future_years = list(range(df_yearly_max['Year'].iloc[-1] + 1,
df_yearly_max['Year'].iloc[-1] + 11))
future_predictions = forecast_future(model_full,
df_yearly_max['Temperature'], 10, scaler)

# Plot
plt.figure(figsize=(10, 4))
plt.plot(df_yearly_max['Year'], df_yearly_max['Temperature'],
label='Historical')
```

```
plt.plot(future_years, future_predictions, label='Forecast (2021–
2030)', linestyle='--', marker='o')
plt.xlabel("Year")
plt.ylabel("Avg Temperature")
plt.title("Yearly Maximum Temperature Forecast (Next 10 Years)")
plt.legend()
plt.tight_layout()
plt.show()

# Print values
forecast_df_max = pd.DataFrame({'Year': future_years, 'Predicted
Temperature': future_predictions})
print("\nForecasted Temperatures (2021–2030):")
print(forecast_df_max)
```

```
Epoch 1/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 5s 5ms/step - loss: 0.1838
Epoch 2/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0912
Epoch 3/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0277
Epoch 4/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0219
Epoch 5/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0216
Epoch 6/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0155
Epoch 7/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0139
Epoch 8/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0129
Epoch 9/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0122
Epoch 10/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0126
Epoch 11/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0136
Epoch 12/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0102
Epoch 13/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0116
Epoch 14/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0122
Epoch 15/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0110
Epoch 16/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0122
Epoch 17/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0139
Epoch 18/50
```

```
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0121
Epoch 19/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0108
Epoch 20/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0118
Epoch 21/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0132
Epoch 22/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0143
Epoch 23/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0126
Epoch 24/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0111
Epoch 25/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0102
Epoch 26/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0106
Epoch 27/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0110
Epoch 28/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0126
Epoch 29/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0129
Epoch 30/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0111
Epoch 31/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0122
Epoch 32/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0138
Epoch 33/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0114
Epoch 34/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0107
Epoch 35/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0136
Epoch 36/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0123
Epoch 37/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0122
Epoch 38/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0112
Epoch 39/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0147
Epoch 40/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0116
Epoch 41/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0129
Epoch 42/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0135
```

```
Epoch 43/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0120
Epoch 44/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0110
Epoch 45/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0125
Epoch 46/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0131
Epoch 47/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0140
Epoch 48/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0120
Epoch 49/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0151
Epoch 50/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0106
```



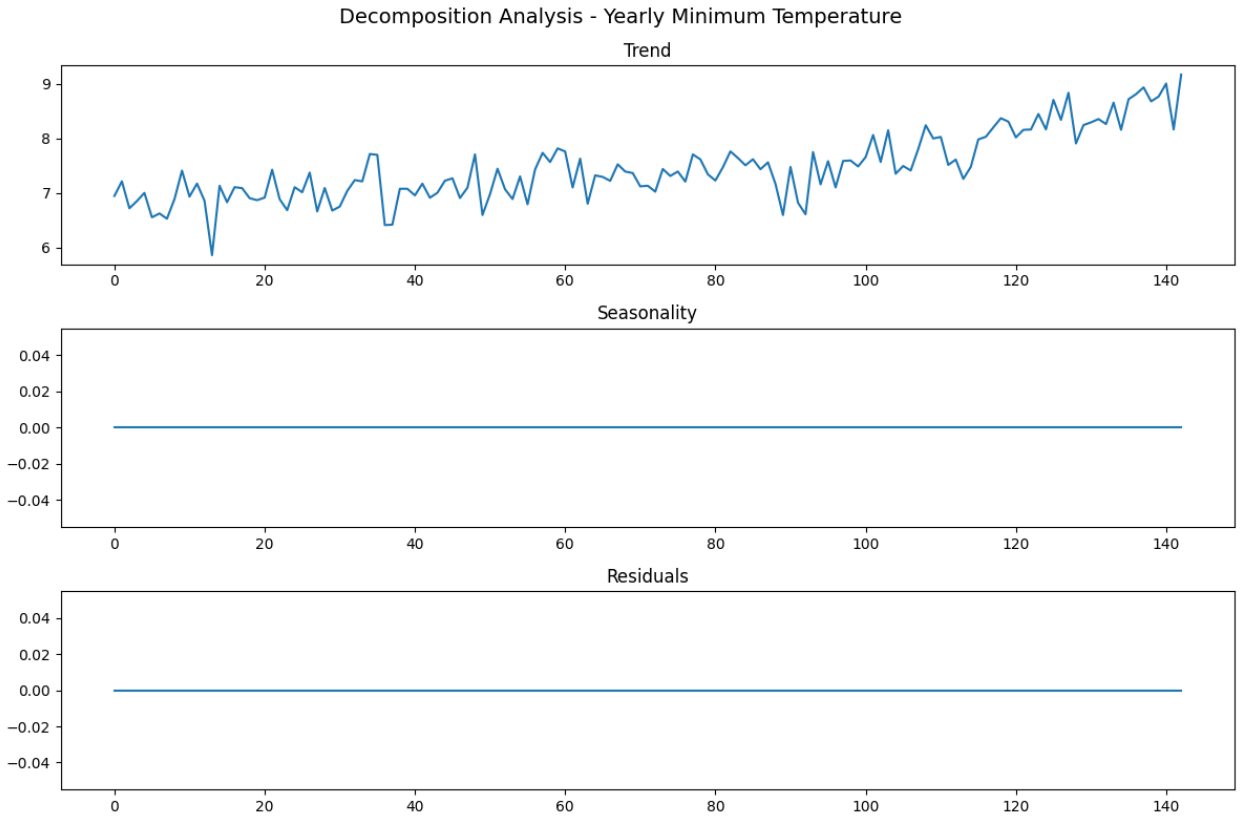

```
Forecasted Temperatures (2021–2030):
   Year  Predicted Temperature
0  2023              10.872462
1  2024              10.988500
2  2025              10.987010
3  2026              10.886519
4  2027              10.990862
5  2028              10.874409
6  2029              10.892254
7  2030              10.871459
8  2031              10.845276
9  2032              10.844179
```

```
#minimum TS
```

```
# Calculate yearly min
df_yearly_min = df.groupby('Year')['Temperature'].min().reset_index()

# Calculate 5-year min
df['5YearGroup'] = (df['Year'] // 5) * 5
df_5year_min = df.groupby('5YearGroup')
['Temperature'].min().reset_index()
df_yearly_min.tail()
```

	Year	Temperature
138	2018	8.676
139	2019	8.764
140	2020	9.002
141	2021	8.162
142	2022	9.166

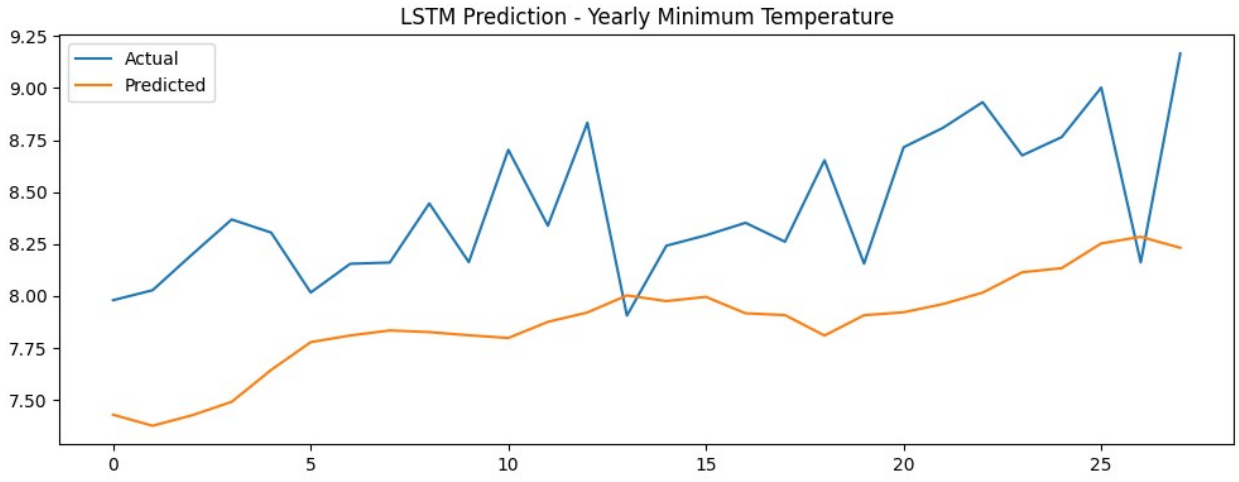
```
# Yearly model
train_lstm_on_series(df_yearly_min['Temperature'], label="Yearly
Minimum Temperature")

# 5-Year model
train_lstm_on_series(df_5year_min['Temperature'], label="5-Year
Minimum Temperature")
```

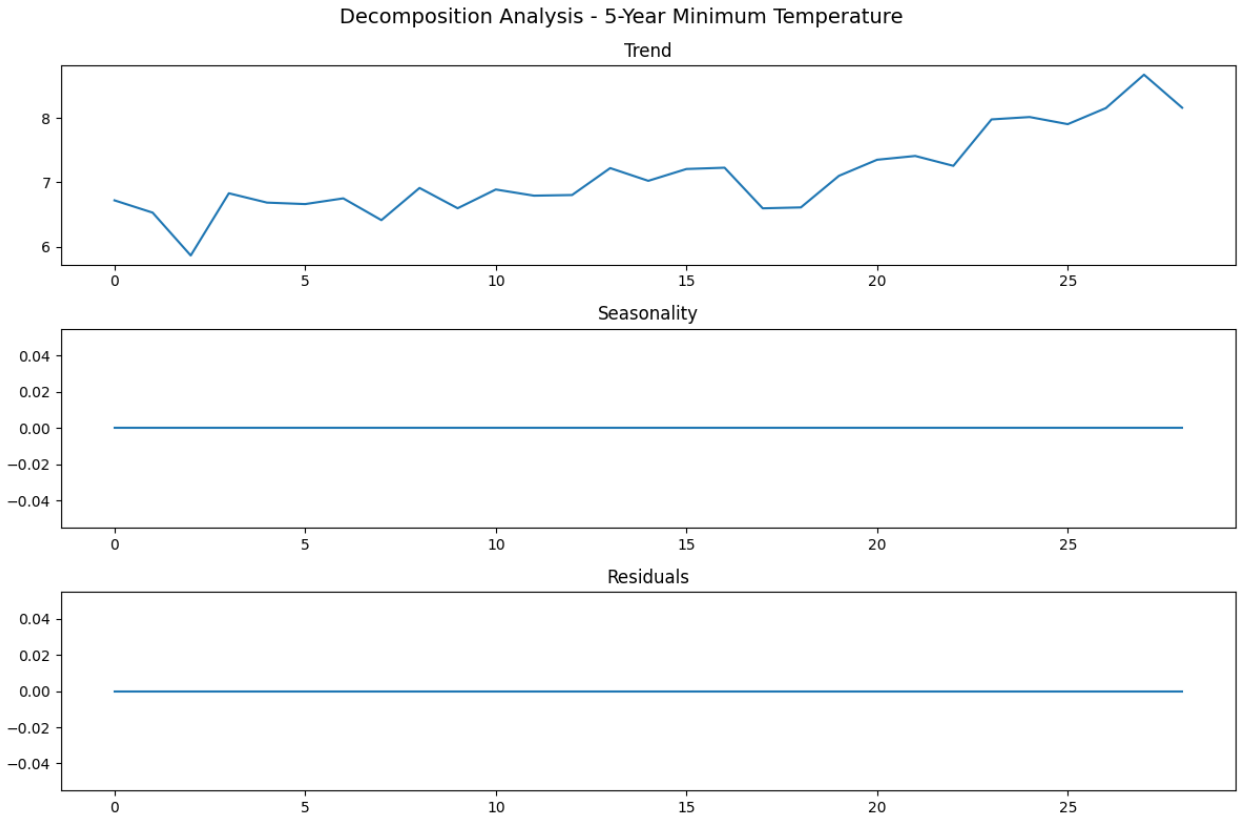


```
Epoch 1/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 5s 5ms/step - loss: 0.1750
Epoch 2/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.1084
Epoch 3/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0191
Epoch 4/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0126
Epoch 5/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0154
Epoch 6/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0134
Epoch 7/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0109
Epoch 8/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0131
Epoch 9/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0140
Epoch 10/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0131
Epoch 11/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0119
Epoch 12/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0154
Epoch 13/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0118
Epoch 14/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0115
Epoch 15/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0119
Epoch 16/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0102
Epoch 17/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0104
Epoch 18/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0124
Epoch 19/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0121
Epoch 20/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0112
Epoch 21/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0113
Epoch 22/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0113
Epoch 23/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0144
Epoch 24/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0108
Epoch 25/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0148
```

```
Epoch 26/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0133
Epoch 27/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0137
Epoch 28/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0129
Epoch 29/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0110
Epoch 30/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0129
Epoch 31/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0115
Epoch 32/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0109
Epoch 33/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0135
Epoch 34/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0120
Epoch 35/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0117
Epoch 36/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0095
Epoch 37/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0128
Epoch 38/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0096
Epoch 39/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0100
Epoch 40/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0135
Epoch 41/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0101
Epoch 42/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0118
Epoch 43/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0134
Epoch 44/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0137
Epoch 45/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0131
Epoch 46/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0091
Epoch 47/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0124
Epoch 48/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0103
Epoch 49/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0118
Epoch 50/50
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0100
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 419ms/step

LSTM Performance (Yearly Minimum Temperature):
MAE: 0.5630
RMSE: 0.6206
```



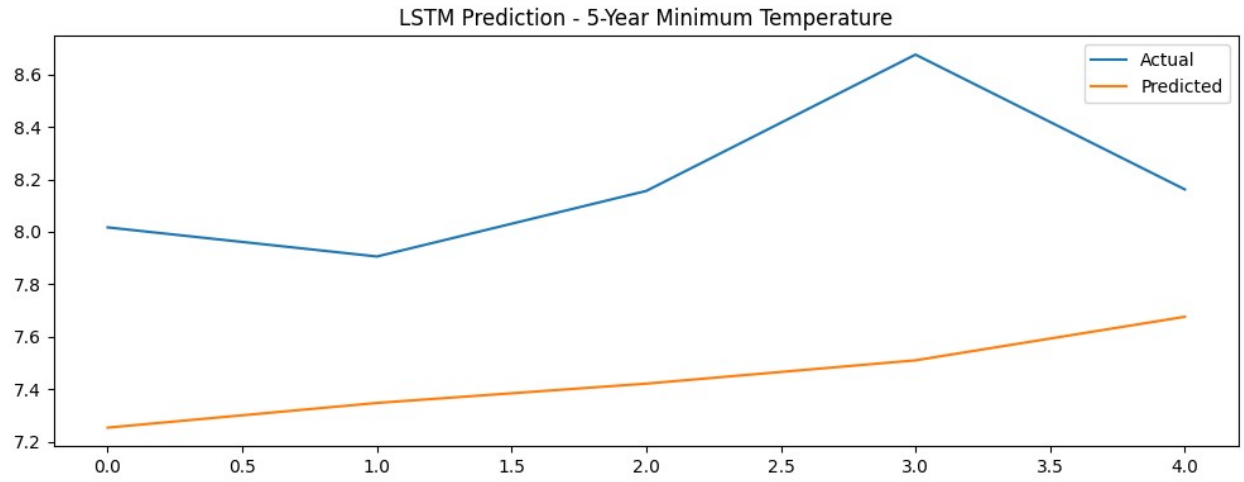



```
Epoch 1/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 9ms/step - loss: 0.1909
Epoch 2/50
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.1558
Epoch 3/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.1475
Epoch 4/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.1236
Epoch 5/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.1185
Epoch 6/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.1015
Epoch 7/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.1024
Epoch 8/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0780
Epoch 9/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0653
Epoch 10/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0356
Epoch 11/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0259
Epoch 12/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0134
Epoch 13/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0149
Epoch 14/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0200
Epoch 15/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0205
Epoch 16/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0161
Epoch 17/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0168
Epoch 18/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0127
Epoch 19/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0128
Epoch 20/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0176
Epoch 21/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0178
Epoch 22/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0157
Epoch 23/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0132
Epoch 24/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0153
Epoch 25/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0115
Epoch 26/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0128
```

```
Epoch 27/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0117
Epoch 28/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0156
Epoch 29/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0136
Epoch 30/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0124
Epoch 31/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0163
Epoch 32/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0126
Epoch 33/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0144
Epoch 34/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0144
Epoch 35/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0150
Epoch 36/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0156
Epoch 37/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0164
Epoch 38/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0136
Epoch 39/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0163
Epoch 40/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0135
Epoch 41/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0146
Epoch 42/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0144
Epoch 43/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0125
Epoch 44/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0137
Epoch 45/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0137
Epoch 46/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0129
Epoch 47/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0129
Epoch 48/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0130
Epoch 49/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0167
Epoch 50/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0124
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 404ms/step
```

```
LSTM Performance (5-Year Minimum Temperature):
MAE: 0.7418
RMSE: 0.7786
```




```
<Sequential name=sequential_52, built=True>
```

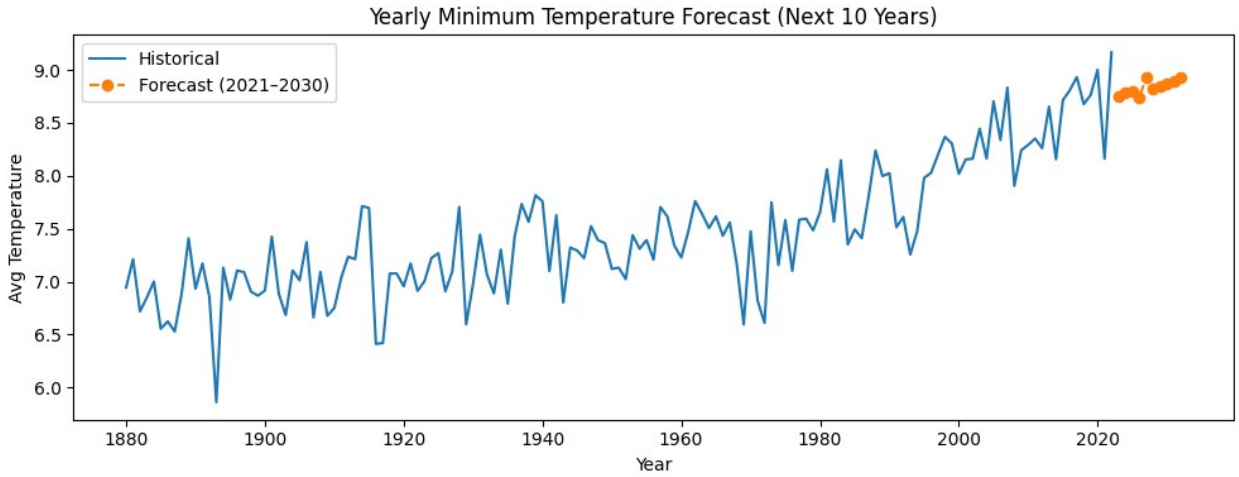
```
# Train on full data for forecasting
scaler = MinMaxScaler()
series_scaled =
scaler.fit_transform(df_yearly_min['Temperature'].values.reshape(-1,
1))

# Re-train model on entire dataset
X_full, y_full = create_sequences(series_scaled, seq_length=5)
X_full = X_full.reshape((X_full.shape[0], X_full.shape[1], 1))
model_full = Sequential([
    LSTM(50, activation='relu', return_sequences=True, input_shape=(5,
1)),
    LSTM(50, activation='relu'),
    Dense(1)
])
model_full.compile(optimizer='adam', loss='mse')
model_full.fit(X_full, y_full, epochs=50, batch_size=8, verbose=1)

# Forecast next 10 years
future_years = list(range(df_yearly_min['Year'].iloc[-1] + 1,
df_yearly_min['Year'].iloc[-1] + 11))
future_predictions = forecast_future(model_full,
df_yearly_min['Temperature'], 10, scaler)

# Plot
plt.figure(figsize=(10, 4))
plt.plot(df_yearly_min['Year'], df_yearly_min['Temperature'],
```

```
label='Historical')
plt.plot(future_years, future_predictions, label='Forecast (2021–
2030)', linestyle='--', marker='o')
plt.xlabel("Year")
plt.ylabel("Avg Temperature")
plt.title("Yearly Minimum Temperature Forecast (Next 10 Years)")
plt.legend()
plt.tight_layout()
plt.show()

# Print values
forecast_df = pd.DataFrame({'Year': future_years, 'Predicted
Temperature': future_predictions})
print("\nForecasted Temperatures (2021–2030):")
print(forecast_df)
```

```
Epoch 1/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 5s 5ms/step - loss: 0.2459
Epoch 2/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.1163
Epoch 3/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0219
Epoch 4/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0172
Epoch 5/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0159
Epoch 6/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0149
Epoch 7/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0169
Epoch 8/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0156
Epoch 9/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0148
Epoch 10/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0152
Epoch 11/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0121
Epoch 12/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0105
Epoch 13/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0137
Epoch 14/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0125
Epoch 15/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0120
Epoch 16/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0137
Epoch 17/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0157
```

```
Epoch 18/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0139
Epoch 19/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0113
Epoch 20/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0134
Epoch 21/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0125
Epoch 22/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0136
Epoch 23/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0168
Epoch 24/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0117
Epoch 25/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0117
Epoch 26/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0117
Epoch 27/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0148
Epoch 28/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0130
Epoch 29/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0121
Epoch 30/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0133
Epoch 31/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0137
Epoch 32/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0116
Epoch 33/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0142
Epoch 34/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0127
Epoch 35/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0126
Epoch 36/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0104
Epoch 37/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0113
Epoch 38/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0157
Epoch 39/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0141
Epoch 40/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0158
Epoch 41/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0125
Epoch 42/50
```

```
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0111
Epoch 43/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0123
Epoch 44/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0130
Epoch 45/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0141
Epoch 46/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0133
Epoch 47/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0162
Epoch 48/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0166
Epoch 49/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0143
Epoch 50/50
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0118
```




```
Forecasted Temperatures (2021–2030):
   Year  Predicted Temperature
0  2023               8.751654
1  2024               8.786212
2  2025               8.795724
3  2026               8.736594
4  2027               8.928153
5  2028               8.822813
6  2029               8.849542
7  2030               8.869045
8  2031               8.889892
9  2032               8.934145
```

```
# Filter data for the last decade
last_decade_df = df[(df['Year'] >= 2010) & (df['Year'] < 2020)].copy()
```

```
# Normalize temperature
scaler = MinMaxScaler(feature_range=(0, 1))
last_decade_df['Temperature_Scaled'] =
scaler.fit_transform(last_decade_df[['Temperature']])

# Prepare sequences
seq_length = 30
data = last_decade_df['Temperature_Scaled'].values

# Use last 30 days for prediction
last_sequence = data[-seq_length:].reshape(1, seq_length, 1)

# Load your last trained model (or retrain here if needed)

# Forecast next 10 days recursively
future_predictions = []

current_sequence = last_sequence.copy()

for _ in range(10):
    next_pred = model.predict(current_sequence)[0, 0]
    future_predictions.append(next_pred)

    # Update the sequence with the new prediction
    current_sequence = np.append(current_sequence[:, 1:, :],
[[[next_pred]]], axis=1)

# Inverse transform predictions to original scale
future_predictions_actual =
scaler.inverse_transform(np.array(future_predictions).reshape(-1, 1))

# Show predictions
for i, temp in enumerate(future_predictions_actual.flatten(),
start=1):
    print(f"Day {i}: Predicted Temp = {temp:.3f}°C")

# Plotting
plt.figure(figsize=(10, 4))
plt.plot(range(1, 11), future_predictions_actual, marker='o',
linestyle='--')
plt.title("LSTM Forecast: Next 10 Days of Temperature")
plt.xlabel("Day")
plt.ylabel("Predicted Temperature (°C)")
plt.grid(True)
plt.tight_layout()
plt.show()

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
import seaborn as sns
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_absolute_error, mean_squared_error
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split

# Set group size to 35 years
group_size = 35
start_year = 1880
end_year = 2020

def adf_test(series):
    result = adfuller(series.dropna())
    print(f"ADF Statistic: {result[0]}")
    print(f"p-value: {result[1]}")
    if result[1] > 0.05:
        print("Non-stationary series (p > 0.05)")
    else:
        print("Stationary series (p <= 0.05)")

def create_sequences(data, seq_length=30):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:i+seq_length])
        y.append(data[i+seq_length])
    return np.array(X), np.array(y)

mae_rmse_results = []

for start in range(start_year, end_year, group_size):
    df_group = df[(df['Year'] >= start) & (df['Year'] < start +
group_size)].copy()
    if df_group.empty:
        continue

    print(f"\nAnalyzing period: {start}-{start + group_size}")

    # Normalize Data
    scaler = MinMaxScaler(feature_range=(0,1))
    df_group['Temperature_Scaled'] =
scaler.fit_transform(df_group[['Temperature']])

    # Prepare Sequences for LSTM
    seq_length = 30
    X, y = create_sequences(df_group['Temperature_Scaled'].values,
```

```
seq_length)

    # Reshape input for LSTM
    X = X.reshape((X.shape[0], X.shape[1], 1))

    # Train-Test Split
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, shuffle=False)

    # Build LSTM Model
    model = Sequential([
        LSTM(50, activation='relu', return_sequences=True,
input_shape=(seq_length, 1)),
        LSTM(50, activation='relu'),
        Dense(1)
    ])

    model.compile(optimizer='adam', loss='mse')

    # Train Model
    model.fit(X_train, y_train, epochs=20, batch_size=16, verbose=1)

    # Predictions
    y_pred_lstm = model.predict(X_test)
    y_pred_lstm = scaler.inverse_transform(y_pred_lstm)

    # Evaluate Performance
    mae_lstm =
mean_absolute_error(scaler.inverse_transform(y_test.reshape(-1,1)),
y_pred_lstm)
    rmse_lstm =
np.sqrt(mean_squared_error(scaler.inverse_transform(y_test.reshape(-
1,1)), y_pred_lstm))

    mae_rmse_results.append([f"{start}-{start + group_size}",
mae_lstm, rmse_lstm])
    print(f'LSTM MAE ({start}-{start + group_size}): {mae_lstm:.4f},
RMSE: {rmse_lstm:.4f}')

# Create DataFrame for results
results_df = pd.DataFrame(mae_rmse_results, columns=["Time Period",
"MAE", "RMSE"])
print(results_df)

# Plot Results
plt.figure(figsize=(12, 6))
plt.plot(results_df["Time Period"], results_df["MAE"], marker='o',
label="MAE")
plt.plot(results_df["Time Period"], results_df["RMSE"], marker='s',
```

```
label="RMSE")
plt.xlabel("Time Period")
plt.ylabel("Error Value")
plt.title("MAE and RMSE for Each 35-Year Group")
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.show()
```

Model MAE (Lower is better) RMSE (Lower is better) Pros Cons ARIMA 0.3038, 0.3772 Works well for small data, interpretable Fails for non-linear patterns LSTM 0.1130, 0.1429 Captures complex relationships, good for long-term forecasting Computationally expensive, needs large data

```
from statsmodels.tsa.seasonal import seasonal_decompose

# Decompose the time series
decomposition = seasonal_decompose(df['Temperature'],
model='additive', period=365)  # Assuming yearly seasonality

# Plot components
fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(14,8))
decomposition.trend.plot(ax=ax1, title='Trend')
decomposition.seasonal.plot(ax=ax2, title='Seasonality')
decomposition.resid.plot(ax=ax3, title='Residuals')
plt.tight_layout()
plt.show()

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_absolute_error, mean_squared_error
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from statsmodels.tsa.seasonal import seasonal_decompose

group_size = 20
start_year = 1880
end_year = 2020

def adf_test(series):
    result = adfuller(series.dropna())
```

```
    print(f"ADF Statistic: {result[0]}")
    print(f"p-value: {result[1]}")
    if result[1] > 0.05:
        print("Non-stationary series (p > 0.05)")
    else:
        print("Stationary series (p <= 0.05)")

def create_sequences(data, seq_length=10):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:i+seq_length])
        y.append(data[i+seq_length])
    return np.array(X), np.array(y)

mae_rmse_results = []

for start in range(start_year, end_year, group_size):
    df_group = df[(df['Year'] >= start) & (df['Year'] < start +
group_size)].copy()
    print(f"Analyzing period: {start}-{start + group_size}")

    # Decomposition
    decomposition = seasonal_decompose(df_group['Temperature'],
model='additive', period=365)
    fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(14,8))
    decomposition.trend.plot(ax=ax1, title='Trend')
    decomposition.seasonal.plot(ax=ax2, title='Seasonality')
    decomposition.resid.plot(ax=ax3, title='Residuals')
    plt.tight_layout()
    plt.show()

    # Normalize Data
    scaler = MinMaxScaler(feature_range=(0,1))
    df_group['Temperature_Scaled'] =
scaler.fit_transform(df_group[['Temperature']])

    # Prepare Sequences for LSTM
    seq_length = 30  # Using 30 days of past data
    X, y = create_sequences(df_group['Temperature_Scaled'].values,
seq_length)

    # Train-Test Split
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, shuffle=False)

    # Build LSTM Model
    model = Sequential([
        LSTM(50, activation='relu', return_sequences=True,
input_shape=(seq_length, 1)),
        LSTM(50, activation='relu'),
```

```
        Dense(1)
    ])

    model.compile(optimizer='adam', loss='mse')

    # Train Model
    model.fit(X_train, y_train, epochs=20, batch_size=16, verbose=1)

    # Predictions
    y_pred_lstm = model.predict(X_test)
    y_pred_lstm = scaler.inverse_transform(y_pred_lstm)

    # Evaluate Performance
    mae_lstm =
mean_absolute_error(scaler.inverse_transform(y_test.reshape(-1,1)),
y_pred_lstm)
    rmse_lstm =
np.sqrt(mean_squared_error(scaler.inverse_transform(y_test.reshape(-
1,1)), y_pred_lstm))

    mae_rmse_results.append([f"{start}-{start + group_size}",
mae_lstm, rmse_lstm])
    print(f'LSTM MAE ({start}-{start + group_size}): {mae_lstm:.4f},
RMSE: {rmse_lstm:.4f}')

# Create DataFrame for results
results_df = pd.DataFrame(mae_rmse_results, columns=["Time Period",
"MAE", "RMSE"])
print(results_df)

# Plot Results
plt.figure(figsize=(12, 6))
plt.plot(results_df["Time Period"], results_df["MAE"], marker='o',
label="MAE")
plt.plot(results_df["Time Period"], results_df["RMSE"], marker='s',
label="RMSE")
plt.xlabel("Time Period")
plt.ylabel("Error Value")
plt.title("MAE and RMSE for Each 10-Year Group")
plt.xticks(rotation=45)
plt.legend()
plt.show()

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima.model import ARIMA
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from statsmodels.tsa.seasonal import seasonal_decompose

group_size = 35
start_year = 1880
end_year = 2020

def adf_test(series):
    result = adfuller(series.dropna())
    print(f"ADF Statistic: {result[0]}")
    print(f"p-value: {result[1]}")
    if result[1] > 0.05:
        print("Non-stationary series (p > 0.05)")
    else:
        print("Stationary series (p <= 0.05)")

def create_sequences(data, seq_length=10):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:i+seq_length])
        y.append(data[i+seq_length])
    return np.array(X), np.array(y)

mae_rmse_results = []

for start in range(start_year, end_year, group_size):
    df_group = df[(df['Year'] >= start) & (df['Year'] < start +
group_size)].copy()
    print(f"Analyzing period: {start}-{start + group_size}")

    # Decomposition
    decomposition = seasonal_decompose(df_group['Temperature'],
model='additive', period=365)
    fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(14,8))
    decomposition.trend.plot(ax=ax1, title='Trend')
    decomposition.seasonal.plot(ax=ax2, title='Seasonality')
    decomposition.resid.plot(ax=ax3, title='Residuals')
    plt.tight_layout()
    plt.show()

    # Normalize Data
    scaler = MinMaxScaler(feature_range=(0,1))
    df_group['Temperature_Scaled'] =
scaler.fit_transform(df_group[['Temperature']])
```

```
    # Prepare Sequences for LSTM
    seq_length = 30  # Using 30 days of past data
    X, y = create_sequences(df_group['Temperature_Scaled'].values,
seq_length)

    # Train-Test Split
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, shuffle=False)

    # Build LSTM Model
    model = Sequential([
        LSTM(50, activation='relu', return_sequences=True,
input_shape=(seq_length, 1)),
        LSTM(50, activation='relu'),
        Dense(1)
    ])

    model.compile(optimizer='adam', loss='mse')

    # Train Model
    model.fit(X_train, y_train, epochs=20, batch_size=16, verbose=1)

    # Predictions
    y_pred_lstm = model.predict(X_test)
    y_pred_lstm = scaler.inverse_transform(y_pred_lstm)

    # Evaluate Performance
    mae_lstm =
mean_absolute_error(scaler.inverse_transform(y_test.reshape(-1,1)),
y_pred_lstm)
    rmse_lstm =
np.sqrt(mean_squared_error(scaler.inverse_transform(y_test.reshape(-
1,1)), y_pred_lstm))

    mae_rmse_results.append([f"{start}-{start + group_size}",
mae_lstm, rmse_lstm])
    print(f'LSTM MAE ({start}-{start + group_size}): {mae_lstm:.4f},
RMSE: {rmse_lstm:.4f}')

# Create DataFrame for results
results_df = pd.DataFrame(mae_rmse_results, columns=["Time Period",
"MAE", "RMSE"])
print(results_df)

# Plot Results
plt.figure(figsize=(12, 6))
plt.plot(results_df["Time Period"], results_df["MAE"], marker='o',
label="MAE")
plt.plot(results_df["Time Period"], results_df["RMSE"], marker='s',
```

```
label="RMSE")
plt.xlabel("Time Period")
plt.ylabel("Error Value")
plt.title("MAE and RMSE for Each 10-Year Group")
plt.xticks(rotation=45)
plt.legend()
plt.show()

from statsmodels.graphics.tsaplots import plot_acf

# ACF Plot
plt.figure(figsize=(10,5))
plot_acf(df['Temperature'], lags=365)
plt.title("Autocorrelation Function (ACF) Plot")
plt.show()



import seaborn as sns

df['Month'] = df.index.month  # Extract month
plt.figure(figsize=(12,6))
sns.boxplot(x='Month', y='Temperature', data=df)
plt.title("Monthly Temperature Variation")
plt.xlabel("Month")
plt.ylabel("Temperature")
plt.show()
```

Forecasting

```
combined_df = results_df.copy()
combined_df['ARIMA_MAE'] = results_arima_df['MAE']
combined_df['ARIMA_RMSE'] = results_arima_df['RMSE']

plt.figure(figsize=(14, 6))
plt.plot(combined_df['Time Period'], combined_df['MAE'], label='LSTM
MAE', marker='o')
plt.plot(combined_df['Time Period'], combined_df['ARIMA_MAE'],
label='ARIMA MAE', marker='s')
plt.plot(combined_df['Time Period'], combined_df['RMSE'], label='LSTM
RMSE', marker='o', linestyle='--')
plt.plot(combined_df['Time Period'], combined_df['ARIMA_RMSE'],
label='ARIMA RMSE', marker='s', linestyle='--')
plt.xticks(rotation=45)
plt.title("Model Performance Comparison: LSTM vs. ARIMA")
plt.xlabel("Time Period")
plt.ylabel("Error")
```

```
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# Fit ARIMA on entire dataset
model_arima = ARIMA(df['Temperature'], order=(5,1,2))
result_arima = model_arima.fit()

# Forecast next 30 days
forecast_arima = result_arima.forecast(steps=30)

# Plot
plt.figure(figsize=(10, 5))
plt.plot(df['Temperature'], label='Original')
plt.plot(pd.date_range(df.index[-1], periods=30, freq='D'),
forecast_arima, label='ARIMA Forecast', color='green')
plt.title("ARIMA Forecast for Next 30 Days")
plt.xlabel("Date")
plt.ylabel("Temperature")
plt.legend()
plt.show()
```